RsCMPX_WlanMeas

Release 5.0.80.4

Rohde & Schwarz

Apr 19, 2024

CONTENTS:

1 Revision History **3**
- 1.1 RsCMPX_WlanMeas . . . 3
  - 1.1.1 Version history . . . 3

2 Getting Started **5**
- 2.1 Introduction . . . 5
- 2.2 Installation . . . 7
- 2.3 Finding Available Instruments . . . 8
- 2.4 Initiating Instrument Session . . . 9
- 2.5 Plain SCPI Communication . . . 12
- 2.6 Error Checking . . . 14
- 2.7 Exception Handling . . . 14
- 2.8 Transferring Files . . . 16
- 2.9 Writing Binary Data . . . 16
- 2.10 Transferring Big Data with Progress . . . 17
- 2.11 Multithreading . . . 18
- 2.12 Logging . . . 21

3 Enums **25**
- 3.1 Bandwidth . . . 25
- 3.2 BurstEvalLength . . . 25
- 3.3 BurstType . . . 25
- 3.4 BurstTypeB . . . 25
- 3.5 CfoEstimation . . . 26
- 3.6 ChannelEstimation . . . 26
- 3.7 Coderate . . . 26
- 3.8 CodingType . . . 26
- 3.9 ConnectorSwitch . . . 26
- 3.10 ConnectorSwitchExt . . . 27
- 3.11 ConnectorTuple . . . 27
- 3.12 DecodeStatus . . . 27
- 3.13 DisplayMode . . . 28
- 3.14 EvmMethod . . . 28
- 3.15 FftOffset . . . 28
- 3.16 FrequencyBand . . . 28
- 3.17 GuardInterval . . . 28
- 3.18 GuiScenario . . . 29
- 3.19 IeeeStandard . . . 29
- 3.20 LowHigh . . . 29
- 3.21 LtfSize . . . 29

3.22 MimoScenario . . . 29
3.23 ModulationFilter . . . 30
3.24 ModulationTypeB . . . 30
3.25 ModulationTypeC . . . 30
3.26 ModulationTypeD . . . 30
3.27 ParameterSetMode . . . 31
3.28 PlcpType . . . 31
3.29 PowerClass . . . 31
3.30 ReceiveMode . . . 31
3.31 RefPower . . . 31
3.32 Repeat . . . 32
3.33 ResourceState . . . 32
3.34 ResultStatus2 . . . 32
3.35 RxConnector . . . 32
3.36 RxConnectorExt . . . 33
3.37 RxConverter . . . 34
3.38 RxTxConverter . . . 34
3.39 SlopeType . . . 34
3.40 StopCondition . . . 35
3.41 SynchroMode . . . 35
3.42 TrainingMode . . . 35
3.43 TriggerSlope . . . 35

4 RepCaps **37**
4.1 Instance (Global) . . . 37
4.2 Antenna . . . 37
4.3 Band . . . 37
4.4 BandwidthA . . . 38
4.5 BandwidthB . . . 38
4.6 BandwidthC . . . 38
4.7 BandwidthD . . . 38
4.8 BandwidthE . . . 38
4.9 BandwidthF . . . 39
4.10 BandwidthG . . . 39
4.11 Channel . . . 39
4.12 Channels . . . 39
4.13 Connector . . . 39
4.14 Mimo . . . 40
4.15 Reserved . . . 40
4.16 ResourceUnit . . . 40
4.17 RxAntenna . . . 41
4.18 Segment . . . 41
4.19 SegmentB . . . 41
4.20 Smi . . . 41
4.21 SMimoPath . . . 41
4.22 Spatial . . . 42
4.23 Stream . . . 42
4.24 TrueMimoPath . . . 42
4.25 User . . . 42
4.26 UserIx . . . 43
4.27 UtError . . . 43

5 Examples **45**

6 RsCMPX_WlanMeas API Structure **47**
6.1 Configure . . . 50
6.1.1 WlanMeas . . . 50
6.1.1.1 Isignal . . . 51
6.1.1.1.1 Dsss . . . 55
6.1.1.1.1.1 Elength . . . 55
6.1.1.1.2 Ofdm . . . 56
6.1.1.1.3 Tdata . . . 56
6.1.1.1.3.1 File . . . 57
6.1.1.2 Mimo . . . 57
6.1.1.3 MultiEval . . . 58
6.1.1.3.1 Compensation . . . 62
6.1.1.3.1.1 EfTaps . . . 63
6.1.1.3.1.2 SkipSymbols . . . 64
6.1.1.3.1.3 Tracking . . . 65
6.1.1.3.2 Demod . . . 66
6.1.1.3.2.1 Fft . . . 67
6.1.1.3.3 Limit . . . 67
6.1.1.3.3.1 Modulation . . . 69
6.1.1.3.3.2 Dsss . . . 69
6.1.1.3.3.3 EhtOfdm . . . 71
6.1.1.3.3.4 CfoDistribution . . . 74
6.1.1.3.3.5 IqOffset . . . 75
6.1.1.3.3.6 Bw<BandwidthG> . . . 75
6.1.1.3.3.7 HeOfdm . . . 77
6.1.1.3.3.8 CfoDistribution . . . 78
6.1.1.3.3.9 EvmAll . . . 79
6.1.1.3.3.10 EvmPilot . . . 83
6.1.1.3.3.11 IqOffset . . . 85
6.1.1.3.3.12 Bw<BandwidthE> . . . 85
6.1.1.3.3.13 HtOfdm . . . 86
6.1.1.3.3.14 IqOffset . . . 89
6.1.1.3.3.15 Bw<BandwidthC> . . . 89
6.1.1.3.3.16 Lofdm . . . 90
6.1.1.3.3.17 Pofdm . . . 93
6.1.1.3.3.18 VhtOfdm . . . 96
6.1.1.3.3.19 IqOffset . . . 99
6.1.1.3.3.20 Bw<BandwidthE> . . . 99
6.1.1.3.3.21 PowerVsTime . . . 100
6.1.1.3.3.22 SpectrFlatness . . . 102
6.1.1.3.3.23 EhtOfdm . . . 102
6.1.1.3.3.24 Bw<BandwidthF> . . . 103
6.1.1.3.3.25 Enable . . . 103
6.1.1.3.3.26 Lower . . . 104
6.1.1.3.3.27 Upper . . . 105
6.1.1.3.3.28 HeOfdm . . . 106
6.1.1.3.3.29 Bw<BandwidthD> . . . 106
6.1.1.3.3.30 Enable . . . 106
6.1.1.3.3.31 Lower . . . 107
6.1.1.3.3.32 Upper . . . 108
6.1.1.3.3.33 HtOfdm . . . 109
6.1.1.3.3.34 Bw<BandwidthC> . . . 109
6.1.1.3.3.35 Enable . . . 110
6.1.1.3.3.36 Lower . . . 111

6.1.1.3.3.37 Upper . . . 112
6.1.1.3.3.38 Lofdm . . . 112
6.1.1.3.3.39 Lower . . . 114
6.1.1.3.3.40 Pofdm . . . 115
6.1.1.3.3.41 Bw<BandwidthB> . . . 115
6.1.1.3.3.42 Enable . . . 115
6.1.1.3.3.43 Lower . . . 116
6.1.1.3.3.44 Upper . . . 117
6.1.1.3.3.45 VhtOfdm . . . 118
6.1.1.3.3.46 Bw<BandwidthE> . . . 118
6.1.1.3.3.47 Enable . . . 119
6.1.1.3.3.48 Lower . . . 120
6.1.1.3.3.49 Upper . . . 121
6.1.1.3.3.50 TsMask . . . 121
6.1.1.3.3.51 Dsss . . . 122
6.1.1.3.3.52 Y . . . 123
6.1.1.3.3.53 EhtOfdm . . . 124
6.1.1.3.3.54 Bw<BandwidthF> . . . 124
6.1.1.3.3.55 AbsLimit . . . 124
6.1.1.3.3.56 Enable . . . 125
6.1.1.3.3.57 Y . . . 126
6.1.1.3.3.58 A . . . 126
6.1.1.3.3.59 B . . . 127
6.1.1.3.3.60 C . . . 128
6.1.1.3.3.61 D . . . 129
6.1.1.3.3.62 HeOfdm . . . 129
6.1.1.3.3.63 Bw<BandwidthD> . . . 130
6.1.1.3.3.64 AbsLimit . . . 130
6.1.1.3.3.65 Enable . . . 131
6.1.1.3.3.66 Y . . . 132
6.1.1.3.3.67 A . . . 132
6.1.1.3.3.68 B . . . 133
6.1.1.3.3.69 C . . . 134
6.1.1.3.3.70 D . . . 135
6.1.1.3.3.71 HtOfdm . . . 135
6.1.1.3.3.72 Bw<BandwidthC> . . . 136
6.1.1.3.3.73 AbsLimit . . . 136
6.1.1.3.3.74 Band<Band> . . . 137
6.1.1.3.3.75 Y . . . 137
6.1.1.3.3.76 A . . . 138
6.1.1.3.3.77 B . . . 139
6.1.1.3.3.78 C . . . 140
6.1.1.3.3.79 D . . . 141
6.1.1.3.3.80 Enable . . . 142
6.1.1.3.3.81 Lofdm . . . 143
6.1.1.3.3.82 Y . . . 143
6.1.1.3.3.83 Pofdm . . . 145
6.1.1.3.3.84 Bw . . . 145
6.1.1.3.3.85 Absolute . . . 146
6.1.1.3.3.86 Y . . . 146
6.1.1.3.3.87 A . . . 146
6.1.1.3.3.88 B . . . 147
6.1.1.3.3.89 C . . . 148
6.1.1.3.3.90 D . . . 149

6.1.1.3.3.91 E . . . 149
6.1.1.3.3.92 F . . . 150
6.1.1.3.3.93 Ca . . . 151
6.1.1.3.3.94 Y . . . 151
6.1.1.3.3.95 A . . . 151
6.1.1.3.3.96 B . . . 152
6.1.1.3.3.97 C . . . 153
6.1.1.3.3.98 D . . . 154
6.1.1.3.3.99 E . . . 155
6.1.1.3.3.100 Cb . . . 155
6.1.1.3.3.101 Y . . . 156
6.1.1.3.3.102 A . . . 156
6.1.1.3.3.103 B . . . 157
6.1.1.3.3.104 C . . . 158
6.1.1.3.3.105 D . . . 158
6.1.1.3.3.106 E . . . 159
6.1.1.3.3.107 Enable . . . 160
6.1.1.3.3.108 UserDefined . . . 161
6.1.1.3.3.109 Y . . . 161
6.1.1.3.3.110 A . . . 161
6.1.1.3.3.111 B . . . 162
6.1.1.3.3.112 C . . . 163
6.1.1.3.3.113 D . . . 164
6.1.1.3.3.114 E . . . 164
6.1.1.3.3.115 VhtOfdm . . . 165
6.1.1.3.3.116 Bw<BandwidthE> . . . 166
6.1.1.3.3.117 AbsLimit . . . 166
6.1.1.3.3.118 Enable . . . 167
6.1.1.3.3.119 Y . . . 168
6.1.1.3.3.120 A . . . 168
6.1.1.3.3.121 B . . . 169
6.1.1.3.3.122 C . . . 170
6.1.1.3.3.123 D . . . 170
6.1.1.3.4 ListPy . . . 171
6.1.1.3.4.1 Result . . . 177
6.1.1.3.4.2 Scount . . . 178
6.1.1.3.4.3 Segment<SegmentB> . . . 179
6.1.1.3.4.4 Bandwidth . . . 180
6.1.1.3.4.5 Btype . . . 181
6.1.1.3.4.6 EnvelopePower . . . 181
6.1.1.3.4.7 Frequency . . . 182
6.1.1.3.4.8 Moffset . . . 183
6.1.1.3.4.9 Mtime . . . 184
6.1.1.3.4.10 Result . . . 185
6.1.1.3.4.11 Rtrigger . . . 186
6.1.1.3.4.12 Scount . . . 187
6.1.1.3.4.13 Setup . . . 188
6.1.1.3.4.14 SingleCmw . . . 189
6.1.1.3.4.15 Connector . . . 189
6.1.1.3.4.16 Standard . . . 190
6.1.1.3.4.17 Stime . . . 191
6.1.1.3.4.18 SingleCmw . . . 192
6.1.1.3.5 PowerVsTime . . . 192
6.1.1.3.6 Result . . . 194

6.1.1.3.7 Scount . . . 199
6.1.1.3.8 SpectrFlatness . . . 200
6.1.1.3.9 TsMask . . . 201
6.1.1.4 RfSettings . . . 203
6.1.1.4.1 Antenna<Antenna> . . . 204
6.1.1.4.2 Eattenuation<Connector> . . . 206
6.1.1.4.3 EnvelopePower<Connector> . . . 207
6.1.1.4.4 Frequency . . . 208
6.1.1.4.4.1 Channels<Channels> . . . 209
6.1.1.4.5 LrStart . . . 211
6.1.1.4.6 Umargin<Connector> . . . 212
6.1.1.5 Smimo . . . 213
6.1.1.6 Tmode . . . 213
6.1.1.6.1 File . . . 214
6.2 Route . . . 215
6.2.1 WlanMeas . . . 215
6.2.1.1 Catalog . . . 216
6.2.1.2 Scenario . . . 217
6.2.1.2.1 Salone . . . 218
6.2.1.2.2 Smi<Smi> . . . 218
6.2.1.2.3 Smimo<SMimoPath> . . . 220
6.2.1.2.4 Tmimo<TrueMimoPath> . . . 221
6.3 Trigger . . . 222
6.3.1 WlanMeas . . . 222
6.3.1.1 MultiEval . . . 222
6.3.1.1.1 Catalog . . . 225
6.3.1.1.1.1 Source . . . 225
6.4 WlanMeas . . . 226
6.4.1 MultiEval . . . 226
6.4.1.1 ListPy . . . 228
6.4.1.1.1 Modulation . . . 228
6.4.1.1.1.1 Bpower . . . 228
6.4.1.1.1.2 Average . . . 229
6.4.1.1.1.3 Current . . . 229
6.4.1.1.1.4 Maximum . . . 230
6.4.1.1.1.5 Minimum . . . 230
6.4.1.1.1.6 StandardDev . . . 231
6.4.1.1.1.7 Cfactor . . . 231
6.4.1.1.1.8 Average . . . 231
6.4.1.1.1.9 Current . . . 232
6.4.1.1.1.10 Maximum . . . 232
6.4.1.1.1.11 Minimum . . . 233
6.4.1.1.1.12 StandardDev . . . 233
6.4.1.1.1.13 CfError . . . 234
6.4.1.1.1.14 Average . . . 234
6.4.1.1.1.15 Current . . . 235
6.4.1.1.1.16 Maximum . . . 235
6.4.1.1.1.17 Minimum . . . 236
6.4.1.1.1.18 StandardDev . . . 236
6.4.1.1.1.19 DcPower . . . 237
6.4.1.1.1.20 Average . . . 237
6.4.1.1.1.21 Current . . . 237
6.4.1.1.1.22 Maximum . . . 238
6.4.1.1.1.23 Minimum . . . 238

6.4.1.1.1.24 StandardDev . . . 239
6.4.1.1.1.25 Dpower . . . 239
6.4.1.1.1.26 Average . . . 240
6.4.1.1.1.27 Current . . . 240
6.4.1.1.1.28 Maximum . . . 241
6.4.1.1.1.29 Minimum . . . 241
6.4.1.1.1.30 StandardDev . . . 242
6.4.1.1.1.31 Dsss . . . 242
6.4.1.1.1.32 Bpower . . . 242
6.4.1.1.1.33 Average . . . 243
6.4.1.1.1.34 Current . . . 243
6.4.1.1.1.35 Maximum . . . 244
6.4.1.1.1.36 Minimum . . . 244
6.4.1.1.1.37 StandardDev . . . 245
6.4.1.1.1.38 CcError . . . 245
6.4.1.1.1.39 Average . . . 245
6.4.1.1.1.40 Current . . . 246
6.4.1.1.1.41 Maximum . . . 246
6.4.1.1.1.42 Minimum . . . 247
6.4.1.1.1.43 StandardDev . . . 247
6.4.1.1.1.44 CfError . . . 248
6.4.1.1.1.45 Average . . . 248
6.4.1.1.1.46 Current . . . 249
6.4.1.1.1.47 Maximum . . . 249
6.4.1.1.1.48 Minimum . . . 250
6.4.1.1.1.49 StandardDev . . . 250
6.4.1.1.1.50 EvmEms . . . 251
6.4.1.1.1.51 Average . . . 251
6.4.1.1.1.52 Current . . . 251
6.4.1.1.1.53 Maximum . . . 252
6.4.1.1.1.54 Minimum . . . 252
6.4.1.1.1.55 StandardDev . . . 253
6.4.1.1.1.56 EvmPeak . . . 253
6.4.1.1.1.57 Average . . . 254
6.4.1.1.1.58 Current . . . 254
6.4.1.1.1.59 Maximum . . . 255
6.4.1.1.1.60 Minimum . . . 255
6.4.1.1.1.61 StandardDev . . . 256
6.4.1.1.1.62 Gimbalance . . . 256
6.4.1.1.1.63 Average . . . 256
6.4.1.1.1.64 Current . . . 257
6.4.1.1.1.65 Maximum . . . 257
6.4.1.1.1.66 Minimum . . . 258
6.4.1.1.1.67 StandardDev . . . 258
6.4.1.1.1.68 IqOffset . . . 259
6.4.1.1.1.69 Average . . . 259
6.4.1.1.1.70 Current . . . 260
6.4.1.1.1.71 Maximum . . . 260
6.4.1.1.1.72 Minimum . . . 261
6.4.1.1.1.73 StandardDev . . . 261
6.4.1.1.1.74 Qerror . . . 262
6.4.1.1.1.75 Average . . . 262
6.4.1.1.1.76 Current . . . 262
6.4.1.1.1.77 Maximum . . . 263

6.4.1.1.1.78 Minimum . . . 263
6.4.1.1.1.79 StandardDev . . . 264
6.4.1.1.1.80 EvmAll . . . 264
6.4.1.1.1.81 Average . . . 265
6.4.1.1.1.82 Current . . . 265
6.4.1.1.1.83 Maximum . . . 266
6.4.1.1.1.84 Minimum . . . 266
6.4.1.1.1.85 StandardDev . . . 267
6.4.1.1.1.86 EvmData . . . 267
6.4.1.1.1.87 Average . . . 267
6.4.1.1.1.88 Current . . . 268
6.4.1.1.1.89 Maximum . . . 268
6.4.1.1.1.90 Minimum . . . 269
6.4.1.1.1.91 StandardDev . . . 269
6.4.1.1.1.92 EvmPilot . . . 270
6.4.1.1.1.93 Average . . . 270
6.4.1.1.1.94 Current . . . 271
6.4.1.1.1.95 Maximum . . . 271
6.4.1.1.1.96 Minimum . . . 272
6.4.1.1.1.97 StandardDev . . . 272
6.4.1.1.1.98 Gimbalance . . . 273
6.4.1.1.1.99 Average . . . 273
6.4.1.1.1.100 Current . . . 273
6.4.1.1.1.101 Maximum . . . 274
6.4.1.1.1.102 Minimum . . . 274
6.4.1.1.1.103 StandardDev . . . 275
6.4.1.1.1.104 IqOffset . . . 275
6.4.1.1.1.105 Average . . . 276
6.4.1.1.1.106 Current . . . 276
6.4.1.1.1.107 Maximum . . . 277
6.4.1.1.1.108 Minimum . . . 277
6.4.1.1.1.109 StandardDev . . . 278
6.4.1.1.1.110 LtfPower . . . 278
6.4.1.1.1.111 Average . . . 278
6.4.1.1.1.112 Current . . . 279
6.4.1.1.1.113 Maximum . . . 279
6.4.1.1.1.114 Minimum . . . 280
6.4.1.1.1.115 StandardDev . . . 280
6.4.1.1.1.116 Pbackoff . . . 281
6.4.1.1.1.117 Ppower . . . 281
6.4.1.1.1.118 Average . . . 282
6.4.1.1.1.119 Current . . . 282
6.4.1.1.1.120 Maximum . . . 283
6.4.1.1.1.121 Minimum . . . 283
6.4.1.1.1.122 StandardDev . . . 284
6.4.1.1.1.123 Qerror . . . 284
6.4.1.1.1.124 Average . . . 284
6.4.1.1.1.125 Current . . . 285
6.4.1.1.1.126 Maximum . . . 285
6.4.1.1.1.127 Minimum . . . 286
6.4.1.1.1.128 StandardDev . . . 286
6.4.1.1.1.129 ScError . . . 287
6.4.1.1.1.130 Average . . . 287
6.4.1.1.1.131 Current . . . 288

6.4.1.1.1.132 Maximum . . . 288
6.4.1.1.1.133 Minimum . . . 289
6.4.1.1.1.134 StandardDev . . . 289
6.4.1.1.1.135 Scount . . . 290
6.4.1.1.2 Segment<SegmentB> . . . 290
6.4.1.1.2.1 Modulation . . . 290
6.4.1.1.2.2 Average . . . 291
6.4.1.1.2.3 Current . . . 292
6.4.1.1.2.4 Dsss . . . 294
6.4.1.1.2.5 Average . . . 294
6.4.1.1.2.6 Current . . . 295
6.4.1.1.2.7 Maximum . . . 296
6.4.1.1.2.8 Minimum . . . 297
6.4.1.1.2.9 StandardDev . . . 298
6.4.1.1.2.10 Maximum . . . 299
6.4.1.1.2.11 Minimum . . . 301
6.4.1.1.2.12 StandardDev . . . 303
6.4.1.1.2.13 TsMask . . . 304
6.4.1.1.2.14 Average . . . 304
6.4.1.1.2.15 Current . . . 305
6.4.1.1.2.16 Frequency . . . 306
6.4.1.1.2.17 Average . . . 306
6.4.1.1.2.18 Current . . . 307
6.4.1.1.2.19 Maximum . . . 308
6.4.1.1.2.20 Minimum . . . 309
6.4.1.1.2.21 Maximum . . . 309
6.4.1.1.2.22 Minimum . . . 310
6.4.1.1.3 Sreliability . . . 311
6.4.1.1.4 TsMask . . . 311
6.4.1.1.4.1 Scount . . . 312
6.4.1.2 Modulation . . . 312
6.4.1.2.1 Acsiso . . . 313
6.4.1.2.1.1 Average . . . 313
6.4.1.2.1.2 Current . . . 315
6.4.1.2.1.3 Maximum . . . 316
6.4.1.2.1.4 StandardDev . . . 318
6.4.1.2.2 Average . . . 320
6.4.1.2.3 CfoDistribution . . . 323
6.4.1.2.4 Cmimo . . . 324
6.4.1.2.4.1 Average . . . 324
6.4.1.2.4.2 Current . . . 325
6.4.1.2.4.3 Maximum . . . 326
6.4.1.2.4.4 Psts . . . 327
6.4.1.2.4.5 Average . . . 327
6.4.1.2.4.6 Current . . . 328
6.4.1.2.4.7 Maximum . . . 329
6.4.1.2.4.8 Minimum . . . 330
6.4.1.2.4.9 StandardDev . . . 331
6.4.1.2.4.10 StandardDev . . . 332
6.4.1.2.5 Current . . . 333
6.4.1.2.6 Dsss . . . 335
6.4.1.2.6.1 Average . . . 336
6.4.1.2.6.2 Current . . . 338
6.4.1.2.6.3 Maximum . . . 340

6.4.1.2.6.4 Minimum . . . 342
6.4.1.2.6.5 StandardDev . . . 344
6.4.1.2.7 EvMagnitude . . . 346
6.4.1.2.7.1 Average . . . 346
6.4.1.2.7.2 Current . . . 347
6.4.1.2.7.3 Maximum . . . 347
6.4.1.2.7.4 StandardDev . . . 348
6.4.1.2.7.5 User<User> . . . 349
6.4.1.2.7.6 Average . . . 349
6.4.1.2.7.7 Current . . . 350
6.4.1.2.7.8 Maximum . . . 350
6.4.1.2.7.9 StandardDev . . . 351
6.4.1.2.7.10 Stream<Stream> . . . 352
6.4.1.2.7.11 Average . . . 352
6.4.1.2.7.12 Current . . . 353
6.4.1.2.7.13 Maximum . . . 354
6.4.1.2.7.14 StandardDev . . . 354
6.4.1.2.8 Maximum . . . 355
6.4.1.2.9 Mimo<Mimo> . . . 358
6.4.1.2.9.1 Average . . . 358
6.4.1.2.9.2 Current . . . 360
6.4.1.2.9.3 Maximum . . . 363
6.4.1.2.9.4 Minimum . . . 365
6.4.1.2.9.5 Segments . . . 367
6.4.1.2.9.6 Average . . . 367
6.4.1.2.9.7 Current . . . 370
6.4.1.2.9.8 Maximum . . . 373
6.4.1.2.9.9 Minimum . . . 376
6.4.1.2.9.10 StandardDev . . . 379
6.4.1.2.9.11 StandardDev . . . 382
6.4.1.2.10 Minimum . . . 384
6.4.1.2.11 Ofdm . . . 387
6.4.1.2.11.1 Average . . . 387
6.4.1.2.11.2 Current . . . 389
6.4.1.2.11.3 Maximum . . . 390
6.4.1.2.11.4 StandardDev . . . 392
6.4.1.2.12 Segments . . . 394
6.4.1.2.12.1 Average . . . 394
6.4.1.2.12.2 Current . . . 397
6.4.1.2.12.3 Maximum . . . 399
6.4.1.2.12.4 Minimum . . . 402
6.4.1.2.12.5 StandardDev . . . 404
6.4.1.2.13 Smimo . . . 407
6.4.1.2.13.1 Average . . . 407
6.4.1.2.13.2 Current . . . 410
6.4.1.2.13.3 Maximum . . . 412
6.4.1.2.13.4 StandardDev . . . 415
6.4.1.2.14 StandardDev . . . 418
6.4.1.3 Ofdma . . . 421
6.4.1.3.1 Info . . . 421
6.4.1.3.2 Uinfo<User> . . . 422
6.4.1.4 Power . . . 423
6.4.1.4.1 Runit<ResourceUnit> . . . 423
6.4.1.4.1.1 Average . . . 423

6.4.1.4.1.2 Current . . . 424
6.4.1.4.1.3 Maximum . . . 424
6.4.1.4.1.4 RxAntenna<RxAntenna> . . . 425
6.4.1.4.1.5 Average . . . 425
6.4.1.4.1.6 Current . . . 426
6.4.1.4.1.7 Maximum . . . 427
6.4.1.4.1.8 StandardDev . . . 427
6.4.1.4.1.9 StandardDev . . . 428
6.4.1.4.2 RxAntenna<RxAntenna> . . . 428
6.4.1.4.2.1 Average . . . 429
6.4.1.4.2.2 Current . . . 429
6.4.1.4.2.3 Maximum . . . 430
6.4.1.4.2.4 StandardDev . . . 430
6.4.1.5 PowerVsTime . . . 431
6.4.1.5.1 FallingEdge . . . 431
6.4.1.5.1.1 Average . . . 431
6.4.1.5.1.2 Current . . . 433
6.4.1.5.1.3 Maximum . . . 434
6.4.1.5.2 Mimo<Mimo> . . . 435
6.4.1.5.2.1 TeDistribution . . . 436
6.4.1.5.3 RisingEdge . . . 437
6.4.1.5.3.1 Average . . . 437
6.4.1.5.3.2 Current . . . 439
6.4.1.5.3.3 Maximum . . . 440
6.4.1.5.4 TeDistribution . . . 441
6.4.1.5.5 Terror . . . 442
6.4.1.5.5.1 Average . . . 443
6.4.1.5.5.2 Current . . . 444
6.4.1.5.5.3 Maximum . . . 445
6.4.1.5.5.4 Mimo<Mimo> . . . 446
6.4.1.5.5.5 Average . . . 446
6.4.1.5.5.6 Current . . . 447
6.4.1.5.5.7 Maximum . . . 449
6.4.1.5.5.8 Minimum . . . 450
6.4.1.5.5.9 StandardDev . . . 451
6.4.1.5.5.10 Minimum . . . 453
6.4.1.5.5.11 StandardDev . . . 454
6.4.1.6 Sinfo . . . 455
6.4.1.6.1 Heb . . . 455
6.4.1.6.1.1 Channel<Channel> . . . 455
6.4.1.6.1.2 Cfield . . . 456
6.4.1.6.1.3 Crc . . . 456
6.4.1.6.1.4 Cru . . . 457
6.4.1.6.1.5 RuAllocation . . . 457
6.4.1.6.1.6 Tail . . . 458
6.4.1.6.1.7 Ufield<UserIx> . . . 459
6.4.1.6.1.8 Coding . . . 459
6.4.1.6.1.9 Crc . . . 460
6.4.1.6.1.10 Dcm . . . 461
6.4.1.6.1.11 Mcs . . . 461
6.4.1.6.1.12 Nsts . . . 462
6.4.1.6.1.13 Reserved . . . 463
6.4.1.6.1.14 SpaConfig . . . 463
6.4.1.6.1.15 StaId . . . 464

6.4.1.6.1.16 Tail . . . 465
6.4.1.6.1.17 TxBeamforming . . . 466
6.4.1.6.2 Hemu . . . 466
6.4.1.6.2.1 Bdcm . . . 467
6.4.1.6.2.2 Bmcs . . . 467
6.4.1.6.2.3 BssColor . . . 468
6.4.1.6.2.4 Bw . . . 468
6.4.1.6.2.5 Crc . . . 469
6.4.1.6.2.6 Doppler . . . 469
6.4.1.6.2.7 GiltfSize . . . 470
6.4.1.6.2.8 Ldpc . . . 470
6.4.1.6.2.9 NltfSymbols . . . 471
6.4.1.6.2.10 NsbSymbols . . . 471
6.4.1.6.2.11 PeDisambiguity . . . 472
6.4.1.6.2.12 PfecPadding . . . 472
6.4.1.6.2.13 Reserved . . . 473
6.4.1.6.2.14 SbCompress . . . 474
6.4.1.6.2.15 SpatialReuse . . . 474
6.4.1.6.2.16 Stbc . . . 475
6.4.1.6.2.17 Tail . . . 475
6.4.1.6.2.18 TxOp . . . 476
6.4.1.6.2.19 UlDl . . . 476
6.4.1.6.3 Hesu . . . 477
6.4.1.6.3.1 BeamChange . . . 477
6.4.1.6.3.2 BssColor . . . 477
6.4.1.6.3.3 Bw . . . 478
6.4.1.6.3.4 Coding . . . 478
6.4.1.6.3.5 Crc . . . 479
6.4.1.6.3.6 Dcm . . . 480
6.4.1.6.3.7 Doppler . . . 480
6.4.1.6.3.8 FormatPy . . . 481
6.4.1.6.3.9 GiltfSize . . . 481
6.4.1.6.3.10 Ldpc . . . 482
6.4.1.6.3.11 Mcs . . . 482
6.4.1.6.3.12 Nsts . . . 483
6.4.1.6.3.13 PeDisambiguity . . . 483
6.4.1.6.3.14 PfecPadding . . . 484
6.4.1.6.3.15 Reserved<Reserved> . . . 484
6.4.1.6.3.16 SpatialReuse . . . 485
6.4.1.6.3.17 Stbc . . . 486
6.4.1.6.3.18 Tail . . . 486
6.4.1.6.3.19 TxBf . . . 487
6.4.1.6.3.20 TxOp . . . 487
6.4.1.6.3.21 UlDl . . . 488
6.4.1.6.4 Hetb . . . 488
6.4.1.6.4.1 BssColor . . . 488
6.4.1.6.4.2 Bw . . . 489
6.4.1.6.4.3 Crc . . . 490
6.4.1.6.4.4 FormatPy . . . 490
6.4.1.6.4.5 Reserved<Reserved> . . . 491
6.4.1.6.4.6 SpatialReuse<Spatial> . . . 492
6.4.1.6.4.7 Tail . . . 493
6.4.1.6.4.8 TxOp . . . 493
6.4.1.6.5 Htsig . . . 494

6.4.1.6.5.1 Aggregation . . . 494
6.4.1.6.5.2 Cbw . . . 495
6.4.1.6.5.3 Crc . . . 495
6.4.1.6.5.4 FecCoding . . . 496
6.4.1.6.5.5 HtLength . . . 496
6.4.1.6.5.6 Mcs . . . 497
6.4.1.6.5.7 Ness . . . 497
6.4.1.6.5.8 Nsounding . . . 498
6.4.1.6.5.9 Reserved . . . 498
6.4.1.6.5.10 ShortGi . . . 499
6.4.1.6.5.11 Smoothing . . . 499
6.4.1.6.5.12 StbCoding . . . 500
6.4.1.6.5.13 Tail . . . 500
6.4.1.6.6 Lsig . . . 501
6.4.1.6.6.1 Length . . . 501
6.4.1.6.6.2 Parity . . . 501
6.4.1.6.6.3 Rate . . . 502
6.4.1.6.6.4 Reserved . . . 503
6.4.1.6.6.5 Tail . . . 503
6.4.1.6.7 VhtSig . . . 504
6.4.1.6.7.1 Beamformed . . . 504
6.4.1.6.7.2 Bw . . . 504
6.4.1.6.7.3 Crc . . . 505
6.4.1.6.7.4 FecCoding . . . 506
6.4.1.6.7.5 Gid . . . 506
6.4.1.6.7.6 Ldpc . . . 507
6.4.1.6.7.7 Paid . . . 507
6.4.1.6.7.8 Reserved<Reserved> . . . 508
6.4.1.6.7.9 Sdisambiguity . . . 509
6.4.1.6.7.10 Sgi . . . 509
6.4.1.6.7.11 Smcs . . . 510
6.4.1.6.7.12 Stbc . . . 510
6.4.1.6.7.13 Sunsts . . . 511
6.4.1.6.7.14 Tail . . . 511
6.4.1.6.7.15 TxOp . . . 512
6.4.1.7 SpectrFlatness . . . 513
6.4.1.7.1 Average . . . 513
6.4.1.7.2 Current . . . 514
6.4.1.7.3 Maximum . . . 516
6.4.1.7.4 Mimo<Mimo> . . . 518
6.4.1.7.4.1 Average . . . 518
6.4.1.7.4.2 Current . . . 520
6.4.1.7.4.3 Maximum . . . 522
6.4.1.7.4.4 Minimum . . . 524
6.4.1.7.4.5 X . . . 526
6.4.1.7.4.6 Average . . . 526
6.4.1.7.4.7 Current . . . 527
6.4.1.7.4.8 Maximum . . . 528
6.4.1.7.4.9 Minimum . . . 530
6.4.1.7.5 Minimum . . . 531
6.4.1.7.6 X . . . 533
6.4.1.7.6.1 Average . . . 533
6.4.1.7.6.2 Current . . . 534
6.4.1.7.6.3 Maximum . . . 535

6.4.1.7.6.4 Minimum . . . 536
6.4.1.8 State . . . 537
6.4.1.8.1 All . . . 537
6.4.1.9 Trace . . . 538
6.4.1.9.1 CfError . . . 538
6.4.1.9.2 EvMagnitude . . . 539
6.4.1.9.2.1 Acsiso . . . 540
6.4.1.9.2.2 Symbol . . . 540
6.4.1.9.2.3 Average . . . 540
6.4.1.9.2.4 Current . . . 541
6.4.1.9.2.5 Maximum . . . 542
6.4.1.9.2.6 Carrier . . . 543
6.4.1.9.2.7 Average . . . 543
6.4.1.9.2.8 Current . . . 545
6.4.1.9.2.9 Maximum . . . 546
6.4.1.9.2.10 Mimo<Mimo> . . . 547
6.4.1.9.2.11 Average . . . 547
6.4.1.9.2.12 Current . . . 549
6.4.1.9.2.13 Maximum . . . 550
6.4.1.9.2.14 Minimum . . . 551
6.4.1.9.2.15 Segment . . . 552
6.4.1.9.2.16 Average . . . 553
6.4.1.9.2.17 Current . . . 554
6.4.1.9.2.18 Maximum . . . 556
6.4.1.9.2.19 Minimum . . . 557
6.4.1.9.2.20 Minimum . . . 558
6.4.1.9.2.21 Segment . . . 559
6.4.1.9.2.22 Average . . . 560
6.4.1.9.2.23 Current . . . 561
6.4.1.9.2.24 Maximum . . . 562
6.4.1.9.2.25 Minimum . . . 564
6.4.1.9.2.26 Dsss . . . 565
6.4.1.9.2.27 Average . . . 565
6.4.1.9.2.28 Current . . . 566
6.4.1.9.2.29 Maximum . . . 567
6.4.1.9.2.30 Nsiso . . . 568
6.4.1.9.2.31 Carrier . . . 569
6.4.1.9.2.32 Average . . . 569
6.4.1.9.2.33 Current . . . 570
6.4.1.9.2.34 Maximum . . . 571
6.4.1.9.2.35 Symbol . . . 572
6.4.1.9.2.36 Average . . . 572
6.4.1.9.2.37 Current . . . 573
6.4.1.9.2.38 Maximum . . . 574
6.4.1.9.2.39 Ofdm . . . 575
6.4.1.9.2.40 Carrier . . . 575
6.4.1.9.2.41 Average . . . 576
6.4.1.9.2.42 Current . . . 577
6.4.1.9.2.43 Maximum . . . 578
6.4.1.9.2.44 Symbol . . . 579
6.4.1.9.2.45 Average . . . 579
6.4.1.9.2.46 Current . . . 580
6.4.1.9.2.47 Maximum . . . 581
6.4.1.9.2.48 Symbol . . . 582

6.4.1.9.2.49 Average . . . 582
6.4.1.9.2.50 Current . . . 583
6.4.1.9.2.51 Maximum . . . 584
6.4.1.9.2.52 Mimo<Mimo> . . . 585
6.4.1.9.2.53 Average . . . 586
6.4.1.9.2.54 Current . . . 587
6.4.1.9.2.55 Maximum . . . 588
6.4.1.9.2.56 Minimum . . . 589
6.4.1.9.2.57 Minimum . . . 591
6.4.1.9.3 IqConstant . . . 592
6.4.1.9.3.1 Inphase . . . 592
6.4.1.9.3.2 Quadrature . . . 593
6.4.1.9.4 PowerVsTime . . . 593
6.4.1.9.4.1 Average . . . 594
6.4.1.9.4.2 Current . . . 595
6.4.1.9.4.3 FallingEdge . . . 596
6.4.1.9.4.4 Average . . . 596
6.4.1.9.4.5 Current . . . 597
6.4.1.9.4.6 Maximum . . . 598
6.4.1.9.4.7 Mimo<Mimo> . . . 599
6.4.1.9.4.8 Average . . . 599
6.4.1.9.4.9 Current . . . 601
6.4.1.9.4.10 Maximum . . . 602
6.4.1.9.4.11 Minimum . . . 603
6.4.1.9.4.12 Segment . . . 604
6.4.1.9.4.13 Average . . . 605
6.4.1.9.4.14 Current . . . 606
6.4.1.9.4.15 Maximum . . . 608
6.4.1.9.4.16 Minimum . . . 609
6.4.1.9.4.17 Time . . . 611
6.4.1.9.4.18 Time . . . 612
6.4.1.9.4.19 Minimum . . . 613
6.4.1.9.4.20 Segment . . . 614
6.4.1.9.4.21 Average . . . 615
6.4.1.9.4.22 Current . . . 616
6.4.1.9.4.23 Maximum . . . 617
6.4.1.9.4.24 Minimum . . . 619
6.4.1.9.4.25 Time . . . 620
6.4.1.9.4.26 Time . . . 621
6.4.1.9.4.27 Maximum . . . 622
6.4.1.9.4.28 Mimo<Mimo> . . . 623
6.4.1.9.4.29 Average . . . 624
6.4.1.9.4.30 Current . . . 625
6.4.1.9.4.31 Maximum . . . 626
6.4.1.9.4.32 Minimum . . . 627
6.4.1.9.4.33 Segment . . . 628
6.4.1.9.4.34 Average . . . 629
6.4.1.9.4.35 Current . . . 630
6.4.1.9.4.36 Maximum . . . 631
6.4.1.9.4.37 Minimum . . . 633
6.4.1.9.4.38 Time . . . 634
6.4.1.9.4.39 Time . . . 636
6.4.1.9.4.40 Minimum . . . 637
6.4.1.9.4.41 RisingEdge . . . 638

6.4.1.9.4.42 Average . . . 638
6.4.1.9.4.43 Current . . . 639
6.4.1.9.4.44 Maximum . . . 640
6.4.1.9.4.45 Mimo<Mimo> . . . 641
6.4.1.9.4.46 Average . . . 642
6.4.1.9.4.47 Current . . . 643
6.4.1.9.4.48 Maximum . . . 644
6.4.1.9.4.49 Minimum . . . 646
6.4.1.9.4.50 Segment . . . 647
6.4.1.9.4.51 Average . . . 647
6.4.1.9.4.52 Current . . . 649
6.4.1.9.4.53 Maximum . . . 650
6.4.1.9.4.54 Minimum . . . 652
6.4.1.9.4.55 Time . . . 653
6.4.1.9.4.56 Time . . . 655
6.4.1.9.4.57 Minimum . . . 656
6.4.1.9.4.58 Segment . . . 657
6.4.1.9.4.59 Average . . . 657
6.4.1.9.4.60 Current . . . 659
6.4.1.9.4.61 Maximum . . . 660
6.4.1.9.4.62 Minimum . . . 661
6.4.1.9.4.63 Time . . . 663
6.4.1.9.4.64 Time . . . 664
6.4.1.9.4.65 Segment . . . 665
6.4.1.9.4.66 Average . . . 665
6.4.1.9.4.67 Current . . . 667
6.4.1.9.4.68 Maximum . . . 668
6.4.1.9.4.69 Minimum . . . 669
6.4.1.9.4.70 Time . . . 670
6.4.1.9.4.71 Time . . . 672
6.4.1.9.5 SpectrFlatness . . . 673
6.4.1.9.5.1 Acarrier . . . 673
6.4.1.9.5.2 Average . . . 673
6.4.1.9.5.3 Current . . . 675
6.4.1.9.5.4 Maximum . . . 676
6.4.1.9.5.5 Minimum . . . 678
6.4.1.9.5.6 Segment . . . 680
6.4.1.9.5.7 Average . . . 680
6.4.1.9.5.8 Current . . . 682
6.4.1.9.5.9 Maximum . . . 684
6.4.1.9.5.10 Minimum . . . 685
6.4.1.9.5.11 Acsiso . . . 687
6.4.1.9.5.12 Average . . . 687
6.4.1.9.5.13 Current . . . 688
6.4.1.9.5.14 Maximum . . . 689
6.4.1.9.5.15 Minimum . . . 690
6.4.1.9.5.16 Average . . . 691
6.4.1.9.5.17 Current . . . 692
6.4.1.9.5.18 Maximum . . . 693
6.4.1.9.5.19 Mimo . . . 693
6.4.1.9.5.20 RxAntenna<RxAntenna> . . . 694
6.4.1.9.5.21 Stream<Stream> . . . 694
6.4.1.9.5.22 Acarrier . . . 695
6.4.1.9.5.23 Average . . . 695

6.4.1.9.5.24 Current . . . 697
6.4.1.9.5.25 Maximum . . . 700
6.4.1.9.5.26 Minimum . . . 702
6.4.1.9.5.27 Average . . . 704
6.4.1.9.5.28 Current . . . 706
6.4.1.9.5.29 Maximum . . . 709
6.4.1.9.5.30 Minimum . . . 711
6.4.1.9.5.31 Segment . . . 713
6.4.1.9.5.32 Acarrier . . . 714
6.4.1.9.5.33 Average . . . 714
6.4.1.9.5.34 Current . . . 716
6.4.1.9.5.35 Maximum . . . 719
6.4.1.9.5.36 Minimum . . . 721
6.4.1.9.5.37 Average . . . 723
6.4.1.9.5.38 Current . . . 725
6.4.1.9.5.39 Maximum . . . 728
6.4.1.9.5.40 Minimum . . . 730
6.4.1.9.5.41 Minimum . . . 732
6.4.1.9.5.42 Ofdm . . . 733
6.4.1.9.5.43 Average . . . 733
6.4.1.9.5.44 Current . . . 734
6.4.1.9.5.45 Maximum . . . 735
6.4.1.9.5.46 Minimum . . . 736
6.4.1.9.5.47 Segment . . . 737
6.4.1.9.5.48 Average . . . 738
6.4.1.9.5.49 Current . . . 739
6.4.1.9.5.50 Maximum . . . 741
6.4.1.9.5.51 Minimum . . . 743
6.4.1.9.6 Terror . . . 745
6.4.1.9.6.1 Mimo<Mimo> . . . 746
6.4.1.9.7 TsMask . . . 747
6.4.1.9.7.1 Average . . . 747
6.4.1.9.7.2 Current . . . 748
6.4.1.9.7.3 Frequency . . . 749
6.4.1.9.7.4 Mask . . . 750
6.4.1.9.7.5 Mimo<Mimo> . . . 752
6.4.1.9.7.6 Segment . . . 753
6.4.1.9.7.7 Segment . . . 755
6.4.1.9.7.8 Maximum . . . 756
6.4.1.9.7.9 Mimo<Mimo> . . . 757
6.4.1.9.7.10 Average . . . 758
6.4.1.9.7.11 Current . . . 759
6.4.1.9.7.12 Maximum . . . 760
6.4.1.9.7.13 Minimum . . . 761
6.4.1.9.7.14 Segment . . . 762
6.4.1.9.7.15 Average . . . 763
6.4.1.9.7.16 Current . . . 764
6.4.1.9.7.17 Maximum . . . 766
6.4.1.9.7.18 Minimum . . . 767
6.4.1.9.7.19 Minimum . . . 768
6.4.1.9.7.20 Segment . . . 769
6.4.1.9.7.21 Average . . . 770
6.4.1.9.7.22 Current . . . 771
6.4.1.9.7.23 Maximum . . . 772

6.4.1.9.7.24 Minimum . . . 773
6.4.1.10 TsMask . . . 774
6.4.1.10.1 Acsiso . . . 775
6.4.1.10.1.1 Average . . . 775
6.4.1.10.1.2 Current . . . 777
6.4.1.10.1.3 Maximum . . . 778
6.4.1.10.2 Average . . . 780
6.4.1.10.3 Current . . . 781
6.4.1.10.4 Dsss . . . 782
6.4.1.10.4.1 Average . . . 783
6.4.1.10.4.2 Current . . . 784
6.4.1.10.4.3 Maximum . . . 785
6.4.1.10.5 Frequency . . . 787
6.4.1.10.5.1 Average . . . 787
6.4.1.10.5.2 Current . . . 788
6.4.1.10.5.3 Maximum . . . 790
6.4.1.10.5.4 Minimum . . . 791
6.4.1.10.6 Maximum . . . 792
6.4.1.10.7 Mimo<Mimo> . . . 794
6.4.1.10.7.1 Average . . . 794
6.4.1.10.7.2 Current . . . 796
6.4.1.10.7.3 Frequency . . . 797
6.4.1.10.7.4 Average . . . 798
6.4.1.10.7.5 Current . . . 799
6.4.1.10.7.6 Maximum . . . 801
6.4.1.10.7.7 Minimum . . . 803
6.4.1.10.7.8 Maximum . . . 804
6.4.1.10.7.9 Minimum . . . 806
6.4.1.10.7.10 Segments . . . 808
6.4.1.10.7.11 Average . . . 808
6.4.1.10.7.12 Current . . . 809
6.4.1.10.7.13 Frequency . . . 811
6.4.1.10.7.14 Average . . . 811
6.4.1.10.7.15 Current . . . 813
6.4.1.10.7.16 Maximum . . . 814
6.4.1.10.7.17 Minimum . . . 816
6.4.1.10.7.18 Maximum . . . 817
6.4.1.10.7.19 Minimum . . . 819
6.4.1.10.8 Minimum . . . 820
6.4.1.10.9 Nsiso . . . 821
6.4.1.10.9.1 Average . . . 822
6.4.1.10.9.2 Current . . . 823
6.4.1.10.9.3 Maximum . . . 825
6.4.1.10.10 Obw . . . 826
6.4.1.10.10.1 Mimo<Mimo> . . . 827
6.4.1.10.10.2 Segments . . . 828
6.4.1.10.10.3 Segments . . . 829
6.4.1.10.11 Ofdm . . . 830
6.4.1.10.11.1 Average . . . 830
6.4.1.10.11.2 Current . . . 832
6.4.1.10.11.3 Maximum . . . 833
6.4.1.10.12 Segments . . . 834
6.4.1.10.12.1 Average . . . 834
6.4.1.10.12.2 Current . . . 835

6.4.1.10.12.3 Frequency . . . 837
6.4.1.10.12.4 Average . . . 837
6.4.1.10.12.5 Current . . . 838
6.4.1.10.12.6 Maximum . . . 839
6.4.1.10.12.7 Minimum . . . 841
6.4.1.10.12.8 Maximum . . . 842
6.4.1.10.12.9 Minimum . . . 843
6.4.1.11 UtError<UtError> . . . 844
6.4.1.11.1 Average . . . 845
6.4.1.11.2 Current . . . 846
6.4.1.11.3 Limit . . . 847
6.4.1.11.4 Margin . . . 848
6.4.1.11.4.1 Average . . . 849
6.4.1.11.4.2 Current . . . 850
6.4.1.11.4.3 Maximum . . . 852
6.4.1.11.4.4 Minimum . . . 853
6.4.1.11.5 Maximum . . . 855
6.4.1.11.6 Minimum . . . 856
6.4.2 Tmode . . . 857
6.4.2.1 Antenna<Antenna> . . . 858
6.4.2.2 Data . . . 859

7 RsCMPX_WlanMeas Utilities 861

8 RsCMPX_WlanMeas Logger 867

9 RsCMPX_WlanMeas Events 869

10 Index 871

Index 873


5G
CMX/P

CHAPTER

ONE

REVISION HISTORY

1.1 RsCMPX_WlanMeas

Rohde & Schwarz CMP180 WLAN Measurement RsCMPX_WlanMeas instrument driver.

Basic Hello-World code:

```
from RsCMPX_WlanMeas import *

instr = RsCMPX_WlanMeas('TCPIP::192.168.2.101::hislip0')
idn = instr.query('*IDN?')
print('Hello, I am: ' + idn)
```

Supported instruments: CMP180

The package is hosted here: https://pypi.org/project/RsCMPX-WlanMeas/

Documentation: https://RsCMPX-WlanMeas.readthedocs.io/

Examples: https://github.com/Rohde-Schwarz/Examples/

1.1.1 Version history

Latest release notes summary: Update for FW 5.0.80

Version 5.0.80

- Update for FW 5.0.80

Version 4.0.151

- Fixed documentation

Version 4.0.150

- First released version for FW 4.0.150

CHAPTER

TWO

GETTING STARTED

2.1 Introduction

RsCMPX_WlanMeas is a Python remote-control communication module for Rohde & Schwarz SCPI-based Test and Measurement Instruments. It represents SCPI commands as fixed APIs and hence provides SCPI autocompletion and helps you to avoid common string typing mistakes.

Basic example of the idea:
SCPI command:
`SYSTem:REFerence:FREQuency:SOURce`
Python module representation:
writing:
`driver.system.reference.frequency.source.set()`
reading:
`driver.system.reference.frequency.source.get()`

Check out this RsCmwBase example:

```
""" Example on how to use the python RsCmw auto-generated instrument driver showing:
- usage of basic properties of the cmw_base object
- basic concept of setting commands and repcaps: DISPlay:WINDow<n>:SELect
- cmw_xxx drivers reliability interface usage
"""

from RsCmwBase import *  # install from pypi.org

RsCmwBase.assert_minimum_version('3.7.90.38')
cmw_base = RsCmwBase('TCPIP::10.112.1.116::INSTR', True, False)
print(f'CMW Base IND: {cmw_base.utilities.idn_string}')
print(f'CMW Instrument options:\n{",".join(cmw_base.utilities.instrument_options)}')
cmw_base.utilities.visa_timeout = 5000

# Sends OPC after each command
cmw_base.utilities.opc_query_after_write = False
```

(continues on next page)

(continued from previous page)

```
# Checks for syst:err? after each command / query
cmw_base.utilities.instrument_status_checking = True

# DISPlay:WINDow<n>:SELect
cmw_base.display.window.select.set(repcap.Window.Win1)
cmw_base.display.window.repcap_window_set(repcap.Window.Win2)
cmw_base.display.window.select.set()

# Self-test
self_test = cmw_base.utilities.self_test()
print(f'CMW self-test result: {self_test} - {"Passed" if self_test[0] == 0 else "Failed"}
↪"')

# Driver's Interface reliability offers a convenient way of reacting on the return value␣
↪Reliability Indicator
cmw_base.reliability.ExceptionOnError = True


# Callback to use for the reliability indicator update event
def my_reliability_handler(event_args: ReliabilityEventArgs):
        print(f'Base Reliability updated.\nContext: {event_args.context}\nMessage:
↪{event_args.message}')


# We register a callback for each change in the reliability indicator
cmw_base.reliability.on_update_handler = my_reliability_handler

# You can obtain the last value of the returned reliability
print(f"\nReliability last value: {cmw_base.reliability.last_value}, context '{cmw_base.
↪reliability.last_context}', message: {cmw_base.reliability.last_message}")

# Reference Frequency Source
cmw_base.system.reference.frequency.set_source(enums.SourceIntExt.INTernal)

# Close the session
cmw_base.close()
```

Couple of reasons why to choose this module over plain SCPI approach:

- Type-safe API using typing module
- You can still use the plain SCPI communication
- You can select which VISA to use or even not use any VISA at all
- Initialization of a new session is straight-forward, no need to set any other properties
- Many useful features are already implemented - reset, self-test, opc-synchronization, error checking, option checking
- Binary data blocks transfer in both directions
- Transfer of arrays of numbers in binary or ASCII format
- File transfers in both directions
- Events generation in case of error, sent data, received data, chunk data (for big files transfer)

- Multithreading session locking - you can use multiple threads talking to one instrument at the same time
- Logging feature tailored for SCPI communication - different for binary and ascii data

2.2 Installation

RsCMPX_WlanMeas is hosted on pypi.org. You can install it with pip (for example, `pip.exe` for Windows), or if you are using Pycharm (and you should be :-) direct in the Pycharm `Packet Management GUI`.

Preconditions

- Installed VISA. You can skip this if you plan to use only socket LAN connection. Download the Rohde & Schwarz VISA for Windows, Linux, Mac OS from here

Option 1 - Installing with pip.exe under Windows

- Start the command console: `WinKey + R`, type `cmd` and hit ENTER
- Change the working directory to the Python installation of your choice (adjust the user name and python version in the path):

```
cd c:\Users\John\AppData\Local\Programs\Python\Python37\Scripts
```

- Install with the command: `pip install RsCMPX_WlanMeas`

Option 2 - Installing in Pycharm

- In Pycharm Menu `File->Settings->Project->Project Interpreter` click on the '+' button on the top left (the last PyCharm version)
- Type `RsCMPX_WlanMeas` in the search box
- If you are behind a Proxy server, configure it in the Menu: `File->Settings->Appearance->System Settings->HTTP Proxy`

For more information about Rohde & Schwarz instrument remote control, check out our Instrument_Remote_Control_Web_Series .

Option 3 - Offline Installation

If you are still reading the installation chapter, it is probably because the options above did not work for you - proxy problems, your boss saw the internet bill... Here are 6 step for installing the RsCMPX_WlanMeas offline:

- Download this python script (**Save target as**): rsinstrument_offline_install.py This installs all the preconditions that the RsCMPX_WlanMeas needs.
- Execute the script in your offline computer (supported is python 3.6 or newer)
- Download the RsCMPX_WlanMeas package to your computer from the pypi.org: https://pypi.org/project/RsCMPX_WlanMeas/#files to for example `c:\temp\`
- Start the command line `WinKey + R`, type `cmd` and hit ENTER
- Change the working directory to the Python installation of your choice (adjust the user name and python version in the path):

```
cd c:\Users\John\AppData\Local\Programs\Python\Python37\Scripts
```

- Install with the command: `pip install c:\temp\RsCMPX_WlanMeas-5.0.80.4.tar`

2.3 Finding Available Instruments

Like the pyvisa's ResourceManager, the RsCMPX_WlanMeas can search for available instruments:

```
"""
Find the instruments in your environment
"""

from RsCMPX_WlanMeas import *

# Use the instr_list string items as resource names in the RsCMPX_WlanMeas constructor
instr_list = RsCMPX_WlanMeas.list_resources("?*")
print(instr_list)
```

If you have more VISAs installed, the one actually used by default is defined by a secret widget called Visa Conflict Manager. You can force your program to use a VISA of your choice:

```
"""
Find the instruments in your environment with the defined VISA implementation
"""

from RsCMPX_WlanMeas import *

# In the optional parameter visa_select you can use for example 'rs' or 'ni'
# Rs Visa also finds any NRP-Zxx USB sensors
instr_list = RsCMPX_WlanMeas.list_resources('?*', 'rs')
print(instr_list)
```

Tip: We believe our R&S VISA is the best choice for our customers. Here are the reasons why:

- Small footprint
- Superior VXI-11 and HiSLIP performance
- Integrated legacy sensors NRP-Zxx support
- Additional VXI-11 and LXI devices search
- Availability for Windows, Linux, Mac OS

2.4 Initiating Instrument Session

RsCMPX_WlanMeas offers four different types of starting your remote-control session. We begin with the most typical case, and progress with more special ones.

Standard Session Initialization

Initiating new instrument session happens, when you instantiate the RsCMPX_WlanMeas object. Below, is a simple Hello World example. Different resource names are examples for different physical interfaces.

```
"""
Simple example on how to use the RsCMPX_WlanMeas module for remote-controlling your␣
↪instrument
Preconditions:

- Installed RsCMPX_WlanMeas Python module Version 5.0.80 or newer from pypi.org
- Installed VISA, for example R&S Visa 5.12 or newer
"""

from RsCMPX_WlanMeas import *

# A good practice is to assure that you have a certain minimum version installed
RsCMPX_WlanMeas.assert_minimum_version('5.0.80')
resource_string_1 = 'TCPIP::192.168.2.101::INSTR'  # Standard LAN connection (also␣
↪called VXI-11)
resource_string_2 = 'TCPIP::192.168.2.101::hislip0'  # Hi-Speed LAN connection - see␣
↪1MA208
resource_string_3 = 'GPIB::20::INSTR'  # GPIB Connection
resource_string_4 = 'USB::0x0AAD::0x0119::022019943::INSTR'  # USB-TMC (Test and␣
↪Measurement Class)

# Initializing the session
driver = RsCMPX_WlanMeas(resource_string_1)

idn = driver.utilities.query_str('*IDN?')
print(f"\nHello, I am: '{idn}'")
print(f'RsCMPX_WlanMeas package version: {driver.utilities.driver_version}')
print(f'Visa manufacturer: {driver.utilities.visa_manufacturer}')
print(f'Instrument full name: {driver.utilities.full_instrument_model_name}')
print(f'Instrument installed options: {",".join(driver.utilities.instrument_options)}')

# Close the session
driver.close()
```

Note: If you are wondering about the missing `ASRL1::INSTR`, yes, it works too, but come on… it's 2023.

Do not care about specialty of each session kind; RsCMPX_WlanMeas handles all the necessary session settings for you. You immediately have access to many identification properties in the interface `driver.utilities` . Here are same of them:

- `idn_string`

- driver_version
- visa_manufacturer
- full_instrument_model_name
- instrument_serial_number
- instrument_firmware_version
- instrument_options

The constructor also contains optional boolean arguments id_query and reset:

```
driver = RsCMPX_WlanMeas('TCPIP::192.168.56.101::hislip0', id_query=True, reset=True)
```

- Setting id_query to True (default is True) checks, whether your instrument can be used with the RsCMPX_WlanMeas module.
- Setting reset to True (default is False) resets your instrument. It is equivalent to calling the reset() method.

Selecting a Specific VISA

Just like in the function list_resources(), the RsCMPX_WlanMeas allows you to choose which VISA to use:

```
"""
Choosing VISA implementation
"""

from RsCMPX_WlanMeas import *

# Force use of the Rs Visa. For NI Visa, use the "SelectVisa='ni'"
driver = RsCMPX_WlanMeas('TCPIP::192.168.56.101::INSTR', True, True, "SelectVisa='rs'")

idn = driver.utilities.query_str('*IDN?')
print(f"\nHello, I am: '{idn}'")
print(f"\nI am using the VISA from: {driver.utilities.visa_manufacturer}")

# Close the session
driver.close()
```

No VISA Session

We recommend using VISA when possible preferrably with HiSlip session because of its low latency. However, if you are a strict VISA denier, RsCMPX_WlanMeas has something for you too - **no Visa installation raw LAN socket**:

```
"""
Using RsCMPX_WlanMeas without VISA for LAN Raw socket communication
"""

from RsCMPX_WlanMeas import *

driver = RsCMPX_WlanMeas('TCPIP::192.168.56.101::5025::SOCKET', True, True, "SelectVisa=
↪'socket'")
print(f'Visa manufacturer: {driver.utilities.visa_manufacturer}')
```

(continues on next page)

(continued from previous page)

```
print(f"\nHello, I am: '{driver.utilities.idn_string}'")

# Close the session
driver.close()
```

Warning: Not using VISA can cause problems by debugging when you want to use the communication Trace Tool. The good news is, you can easily switch to use VISA and back just by changing the constructor arguments. The rest of your code stays unchanged.

Simulating Session

If a colleague is currently occupying your instrument, leave him in peace, and open a simulating session:

```
driver = RsCMPX_WlanMeas('TCPIP::192.168.56.101::hislip0', True, True, "Simulate=True")
```

More `option_string` tokens are separated by comma:

```
driver = RsCMPX_WlanMeas('TCPIP::192.168.56.101::hislip0', True, True, "SelectVisa='rs',␣
↪Simulate=True")
```

Shared Session

In some scenarios, you want to have two independent objects talking to the same instrument. Rather than opening a second VISA connection, share the same one between two or more RsCMPX_WlanMeas objects:

```
"""
Sharing the same physical VISA session by two different RsCMPX_WlanMeas objects
"""

from RsCMPX_WlanMeas import *

driver1 = RsCMPX_WlanMeas('TCPIP::192.168.56.101::INSTR', True, True)
driver2 = RsCMPX_WlanMeas.from_existing_session(driver1)

print(f'driver1: {driver1.utilities.idn_string}')
print(f'driver2: {driver2.utilities.idn_string}')

# Closing the driver2 session does not close the driver1 session - driver1 is the
↪'session master'
driver2.close()
print(f'driver2: I am closed now')

print(f'driver1: I am  still opened and working: {driver1.utilities.idn_string}')
driver1.close()
print(f'driver1: Only now I am closed.')
```

Note: The `driver1` is the object holding the 'master' session. If you call the `driver1.close()`, the `driver2` loses its instrument session as well, and becomes pretty much useless.

2.5 Plain SCPI Communication

After you have opened the session, you can use the instrument-specific part described in the RsCMPX_WlanMeas API Structure. If for any reason you want to use the plain SCPI, use the `utilities` interface's two basic methods:

- `write_str()` - writing a command without an answer, for example ***RST**
- `query_str()` - querying your instrument, for example the ***IDN?** query

You may ask a question. Actually, two questions:

- **Q1**: Why there are not called `write()` and `query()` ?
- **Q2**: Where is the `read()` ?

Answer 1: Actually, there are - the `write_str()` / `write()` and `query_str()` / `query()` are aliases, and you can use any of them. We promote the `_str` names, to clearly show you want to work with strings. Strings in Python3 are Unicode, the *bytes* and *string* objects are not interchangeable, since one character might be represented by more than 1 byte. To avoid mixing string and binary communication, all the method names for binary transfer contain `_bin` in the name.

Answer 2: Short answer - you do not need it. Long answer - your instrument never sends unsolicited responses. If you send a set command, you use `write_str()`. For a query command, you use `query_str()`. So, you really do not need it...

Bottom line - if you are used to `write()` and `query()` methods, from pyvisa, the `write_str()` and `query_str()` are their equivalents.

Enough with the theory, let us look at an example. Simple write, and query:

```
"""
Basic string write_str / query_str
"""

from RsCMPX_WlanMeas import *

driver = RsCMPX_WlanMeas('TCPIP::192.168.56.101::INSTR')
driver.utilities.write_str('*RST')
response = driver.utilities.query_str('*IDN?')
print(response)

# Close the session
driver.close()
```

This example is so-called "*University-Professor-Example*" - good to show a principle, but never used in praxis. The abovementioned commands are already a part of the driver's API. Here is another example, achieving the same goal:

```
"""
Basic string write_str / query_str
"""

from RsCMPX_WlanMeas import *

driver = RsCMPX_WlanMeas('TCPIP::192.168.56.101::INSTR')
driver.utilities.reset()
print(driver.utilities.idn_string)
```

(continues on next page)

(continued from previous page)

```
# Close the session
driver.close()
```

One additional feature we need to mention here: **VISA timeout**. To simplify, VISA timeout plays a role in each `query_xxx()`, where the controller (your PC) has to prevent waiting forever for an answer from your instrument. VISA timeout defines that maximum waiting time. You can set/read it with the `visa_timeout` property:

```
# Timeout in milliseconds
driver.utilities.visa_timeout = 3000
```

After this time, the RsCMPX_WlanMeas raises an exception. Speaking of exceptions, an important feature of the RsCMPX_WlanMeas is **Instrument Status Checking**. Check out the next chapter that describes the error checking in details.

For completion, we mention other string-based `write_xxx()` and `query_xxx()` methods - all in one example. They are convenient extensions providing type-safe float/boolean/integer setting/querying features:

```
"""
Basic string write_xxx / query_xxx
"""

from RsCMPX_WlanMeas import *

driver = RsCMPX_WlanMeas('TCPIP::192.168.56.101::INSTR')
driver.utilities.visa_timeout = 5000
driver.utilities.instrument_status_checking = True
driver.utilities.write_int('SWEEP:COUNT ', 10)  # sending 'SWEEP:COUNT 10'
driver.utilities.write_bool('SOURCE:RF:OUTPUT:STATE ', True)  # sending
↪'SOURCE:RF:OUTPUT:STATE ON'
driver.utilities.write_float('SOURCE:RF:FREQUENCY ', 1E9)  # sending 'SOURCE:RF:FREQUENCY␣
↪1000000000'

sc = driver.utilities.query_int('SWEEP:COUNT?')  # returning integer number sc=10
out = driver.utilities.query_bool('SOURCE:RF:OUTPUT:STATE?')  # returning boolean␣
↪out=True
freq = driver.utilities.query_float('SOURCE:RF:FREQUENCY?')  # returning float number␣
↪freq=1E9

# Close the session
driver.close()
```

Lastly, a method providing basic synchronization: `query_opc()`. It sends query ***OPC?** to your instrument. The instrument waits with the answer until all the tasks it currently has in a queue are finished. This way your program waits too, and this way it is synchronized with the actions in the instrument. Remember to have the VISA timeout set to an appropriate value to prevent the timeout exception. Here's the snippet:

```
driver.utilities.visa_timeout = 3000
driver.utilities.write_str("INIT")
driver.utilities.query_opc()

# The results are ready now to fetch
results = driver.utilities.query_str("FETCH:MEASUREMENT?")
```

Tip: Wait, there's more: you can send the ***OPC?** after each `write_xxx()` automatically:

```
# Default value after init is False
driver.utilities.opc_query_after_write = True
```

2.6 Error Checking

RsCMPX_WlanMeas pushes limits even further (internal R&S joke): It has a built-in mechanism that after each command/query checks the instrument's status subsystem, and raises an exception if it detects an error. For those who are already screaming: **Speed Performance Penalty!!!**, don't worry, you can disable it.

Instrument status checking is very useful since in case your command/query caused an error, you are immediately informed about it. Status checking has in most cases no practical effect on the speed performance of your program. However, if for example, you do many repetitions of short write/query sequences, it might make a difference to switch it off:

```
# Default value after init is True
driver.utilities.instrument_status_checking = False
```

To clear the instrument status subsystem of all errors, call this method:

```
driver.utilities.clear_status()
```

Instrument's status system error queue is clear-on-read. It means, if you query its content, you clear it at the same time. To query and clear list of all the current errors, use this snippet:

```
errors_list = driver.utilities.query_all_errors()
```

See the next chapter on how to react on errors.

2.7 Exception Handling

The base class for all the exceptions raised by the RsCMPX_WlanMeas is `RsInstrException`. Inherited exception classes:

- `ResourceError` raised in the constructor by problems with initiating the instrument, for example wrong or non-existing resource name
- `StatusException` raised if a command or a query generated error in the instrument's error queue
- `TimeoutException` raised if a visa timeout or an opc timeout is reached

In this example we show usage of all of them. Because it is difficult to generate an error using the instrument-specific SCPI API, we use plain SCPI commands:

```
"""
Showing how to deal with exceptions
"""

from RsCMPX_WlanMeas import *

```

(continues on next page)

(continued from previous page)

```
driver = None
# Try-catch for initialization. If an error occures, the ResourceError is raised
try:
    driver = RsCMPX_WlanMeas('TCPIP::10.112.1.179::hislip0')
except ResourceError as e:
    print(e.args[0])
    print('Your instrument is probably OFF...')
    # Exit now, no point of continuing
    exit(1)

# Dealing with commands that potentially generate errors OPTION 1:
# Switching the status checking OFF termporarily
driver.utilities.instrument_status_checking = False
driver.utilities.write_str('MY:MISSpelled:COMMand')
# Clear the error queue
driver.utilities.clear_status()
# Status checking ON again
driver.utilities.instrument_status_checking = True

# Dealing with queries that potentially generate errors OPTION 2:
try:
    # You migh want to reduce the VISA timeout to avoid long waiting
    driver.utilities.visa_timeout = 1000
    driver.utilities.query_str('MY:WRONg:QUERy?')

except StatusException as e:
    # Instrument status error
    print(e.args[0])
    print('Nothing to see here, moving on...')

except TimeoutException as e:
    # Timeout error
    print(e.args[0])
    print('That took a long time...')

except RsInstrException as e:
    # RsInstrException is a base class for all the RsCMPX_WlanMeas exceptions
    print(e.args[0])
    print('Some other RsCMPX_WlanMeas error...')

finally:
    driver.utilities.visa_timeout = 5000
    # Close the session in any case
    driver.close()
```

Tip: General rules for exception handling:

- If you are sending commands that might generate errors in the instrument, for example deleting a file which does not exist, use the **OPTION 1** - temporarily disable status checking, send the command, clear the error queue and enable the status checking again.
- If you are sending queries that might generate errors or timeouts, for example querying measurement that can not be performed at the moment, use the **OPTION 2** - try/except with optionally adjusting the timeouts.

2.8 Transferring Files

Instrument -> PC

You definitely experienced it: you just did a perfect measurement, saved the results as a screenshot to an instrument's storage drive. Now you want to transfer it to your PC. With RsCMPX_WlanMeas, no problem, just figure out where the screenshot was stored on the instrument. In our case, it is */var/user/instr_screenshot.png*:

```
driver.utilities.read_file_from_instrument_to_pc(
    r'/var/user/instr_screenshot.png',
    r'c:\temp\pc_screenshot.png')
```

PC -> Instrument

Another common scenario: Your cool test program contains a setup file you want to transfer to your instrument: Here is the RsCMPX_WlanMeas one-liner split into 3 lines:

```
driver.utilities.send_file_from_pc_to_instrument(
    r'c:\MyCoolTestProgram\instr_setup.sav',
    r'/var/appdata/instr_setup.sav')
```

2.9 Writing Binary Data

Writing from bytes

An example where you need to send binary data is a waveform file of a vector signal generator. First, you compose your wform_data as `bytes`, and then you send it with `write_bin_block()`:

```
# MyWaveform.wv is an instrument file name under which this data is stored
driver.utilities.write_bin_block(
    "SOUR:BB:ARB:WAV:DATA 'MyWaveform.wv',",
    wform_data)
```

Note: Notice the `write_bin_block()` has two parameters:

- `string` parameter `cmd` for the SCPI command
- `bytes` parameter `payload` for the actual binary data to send

Writing from PC files

Similar to querying binary data to a file, you can write binary data from a file. The second parameter is then the PC file path the content of which you want to send:

```
driver.utilities.write_bin_block_from_file(
    "SOUR:BB:ARB:WAV:DATA 'MyWaveform.wv',",
    r"c:\temp\wform_data.wv")
```

2.10 Transferring Big Data with Progress

We can agree that it can be annoying using an application that shows no progress for long-lasting operations. The same is true for remote-control programs. Luckily, the RsCMPX_WlanMeas has this covered. And, this feature is quite universal - not just for big files transfer, but for any data in both directions.

RsCMPX_WlanMeas allows you to register a function (programmers fancy name is `callback`), which is then periodicaly invoked after transfer of one data chunk. You can define that chunk size, which gives you control over the callback invoke frequency. You can even slow down the transfer speed, if you want to process the data as they arrive (direction instrument -> PC).

To show this in praxis, we are going to use another *University-Professor-Example*: querying the ***IDN?** with chunk size of 2 bytes and delay of 200ms between each chunk read:

```
"""
Event handlers by reading
"""

from RsCMPX_WlanMeas import *
import time


def my_transfer_handler(args):
    """Function called each time a chunk of data is transferred"""
    # Total size is not always known at the beginning of the transfer
    total_size = args.total_size if args.total_size is not None else "unknown"

    print(f"Context: '{args.context}{'with opc' if args.opc_sync else ''}', "
        f"chunk {args.chunk_ix}, "
        f"transferred {args.transferred_size} bytes, "
        f"total size {total_size}, "
        f"direction {'reading' if args.reading else 'writing'}, "
        f"data '{args.data}'")

    if args.end_of_transfer:
        print('End of Transfer')
    time.sleep(0.2)


driver = RsCMPX_WlanMeas('TCPIP::192.168.56.101::INSTR')

driver.events.on_read_handler = my_transfer_handler
# Switch on the data to be included in the event arguments
```

(continues on next page)

(continued from previous page)

```
# The event arguments args.data will be updated
driver.events.io_events_include_data = True
# Set data chunk size to 2 bytes
driver.utilities.data_chunk_size = 2
driver.utilities.query_str('*IDN?')
# Unregister the event handler
driver.utilities.on_read_handler = None

# Close the session
driver.close()
```

If you start it, you might wonder (or maybe not): why is the `args.total_size = None`? The reason is, in this particular case the RsCMPX_WlanMeas does not know the size of the complete response up-front. However, if you use the same mechanism for transfer of a known data size (for example, file transfer), you get the information about the total size too, and hence you can calculate the progress as:

*progress [pct] = 100 * args.transferred_size / args.total_size*

Snippet of transferring file from PC to instrument, the rest of the code is the same as in the previous example:

```
driver.events.on_write_handler = my_transfer_handler
driver.events.io_events_include_data = True
driver.data_chunk_size = 1000
driver.utilities.send_file_from_pc_to_instrument(
    r'c:\MyCoolTestProgram\my_big_file.bin',
    r'/var/user/my_big_file.bin')
# Unregister the event handler
driver.events.on_write_handler = None
```

2.11 Multithreading

You are at the party, many people talking over each other. Not every person can deal with such crosstalk, neither can measurement instruments. For this reason, RsCMPX_WlanMeas has a feature of scheduling the access to your instrument by using so-called **Locks**. Locks make sure that there can be just one client at a time *talking* to your instrument. Talking in this context means completing one communication step - one command write or write/read or write/read/error check.

To describe how it works, and where it matters, we take three typical mulithread scenarios:

One instrument session, accessed from multiple threads

You are all set - the lock is a part of your instrument session. Check out the following example - it will execute properly, although the instrument gets 10 queries at the same time:

```
"""
Multiple threads are accessing one RsCMPX_WlanMeas object
"""

import threading
from RsCMPX_WlanMeas import *
```

(continues on next page)

(continued from previous page)

```
def execute(session):
    """Executed in a separate thread."""
    session.utilities.query_str('*IDN?')


driver = RsCMPX_WlanMeas('TCPIP::192.168.56.101::INSTR')
threads = []
for i in range(10):
    t = threading.Thread(target=execute, args=(driver, ))
    t.start()
    threads.append(t)
print('All threads started')

# Wait for all threads to join this main thread
for t in threads:
    t.join()
print('All threads ended')

driver.close()
```

Shared instrument session, accessed from multiple threads

Same as the previous case, you are all set. The session carries the lock with it. You have two objects, talking to the same instrument from multiple threads. Since the instrument session is shared, the same lock applies to both objects causing the exclusive access to the instrument.

Try the following example:

```
"""
Multiple threads are accessing two RsCMPX_WlanMeas objects with shared session
"""

import threading
from RsCMPX_WlanMeas import *


def execute(session: RsCMPX_WlanMeas, session_ix, index) -> None:
    """Executed in a separate thread."""
    print(f'{index} session {session_ix} query start...')
    session.utilities.query_str('*IDN?')
    print(f'{index} session {session_ix} query end')


driver1 = RsCMPX_WlanMeas('TCPIP::192.168.56.101::INSTR')
driver2 = RsCMPX_WlanMeas.from_existing_session(driver1)
driver1.utilities.visa_timeout = 200
driver2.utilities.visa_timeout = 200
# To see the effect of crosstalk, uncomment this line
# driver2.utilities.clear_lock()
```

(continues on next page)

(continued from previous page)

```
threads = []
for i in range(10):
    t = threading.Thread(target=execute, args=(driver1, 1, i,))
    t.start()
    threads.append(t)
    t = threading.Thread(target=execute, args=(driver2, 2, i,))
    t.start()
    threads.append(t)
print('All threads started')

# Wait for all threads to join this main thread
for t in threads:
    t.join()
print('All threads ended')

driver2.close()
driver1.close()
```

As you see, everything works fine. If you want to simulate some party crosstalk, uncomment the line `driver2.utilities.clear_lock()`. Thich causes the driver2 session lock to break away from the driver1 session lock. Although the driver1 still tries to schedule its instrument access, the driver2 tries to do the same at the same time, which leads to all the fun stuff happening.

Multiple instrument sessions accessed from multiple threads

Here, there are two possible scenarios depending on the instrument's VISA interface:

- Your are lucky, because you instrument handles each remote session completely separately. An example of such instrument is SMW200A. In this case, you have no need for session locking.
- Your instrument handles all sessions with one set of in/out buffers. You need to lock the session for the duration of a talk. And you are lucky again, because the RsCMPX_WlanMeas takes care of it for you. The text below describes this scenario.

Run the following example:

```
"""
Multiple threads are accessing two RsCMPX_WlanMeas objects with two separate sessions
"""

import threading
from RsCMPX_WlanMeas import *


def execute(session: RsCMPX_WlanMeas, session_ix, index) -> None:
    """Executed in a separate thread."""
    print(f'{index} session {session_ix} query start...')
    session.utilities.query_str('*IDN?')
    print(f'{index} session {session_ix} query end')


driver1 = RsCMPX_WlanMeas('TCPIP::192.168.56.101::INSTR')
driver2 = RsCMPX_WlanMeas('TCPIP::192.168.56.101::INSTR')
```

(continues on next page)

(continued from previous page)

```
driver1.utilities.visa_timeout = 200
driver2.utilities.visa_timeout = 200

# Synchronise the sessions by sharing the same lock
driver2.utilities.assign_lock(driver1.utilities.get_lock())  # To see the effect of␣
↪crosstalk, comment this line

threads = []
for i in range(10):
    t = threading.Thread(target=execute, args=(driver1, 1, i,))
    t.start()
    threads.append(t)
    t = threading.Thread(target=execute, args=(driver2, 2, i,))
    t.start()
    threads.append(t)
print('All threads started')

# Wait for all threads to join this main thread
for t in threads:
    t.join()
print('All threads ended')

driver2.close()
driver1.close()
```

You have two completely independent sessions that want to talk to the same instrument at the same time. This will not go well, unless they share the same session lock. The key command to achieve this is `driver2.utilities.assign_lock(driver1.utilities.get_lock())` Try to comment it and see how it goes. If despite commenting the line the example runs without issues, you are lucky to have an instrument similar to the SMW200A.

2.12 Logging

Yes, the logging again. This one is tailored for instrument communication. You will appreciate such handy feature when you troubleshoot your program, or just want to protocol the SCPI communication for your test reports.

What can you actually do with the logger?

- Write SCPI communication to a stream-like object, for example console or file, or both simultaneously
- Log only errors and skip problem-free parts; this way you avoid going through thousands lines of texts
- Investigate duration of certain operations to optimize your program's performance
- Log custom messages from your program

Let us take this basic example:

```
"""
Basic logging example to the console
"""

from RsCMPX_WlanMeas import *
```

(continues on next page)

(continued from previous page)

```
driver = RsCMPX_WlanMeas('TCPIP::192.168.1.101::INSTR')

# Switch ON logging to the console.
driver.utilities.logger.log_to_console = True
driver.utilities.logger.mode = LoggingMode.On
driver.utilities.reset()

# Close the session
driver.close()
```

Console output:

```
10:29:10.819     TCPIP::192.168.1.101::INSTR     0.976 ms  Write: *RST
10:29:10.819     TCPIP::192.168.1.101::INSTR  1884.985 ms  Status check: OK
10:29:12.704     TCPIP::192.168.1.101::INSTR     0.983 ms  Query OPC: 1
10:29:12.705     TCPIP::192.168.1.101::INSTR     2.892 ms  Clear status: OK
10:29:12.708     TCPIP::192.168.1.101::INSTR     3.905 ms  Status check: OK
10:29:12.712     TCPIP::192.168.1.101::INSTR     1.952 ms  Close: Closing session
```

The columns of the log are aligned for better reading. Columns meaning:

- (1) Start time of the operation
- (2) Device resource name (you can set an alias)
- (3) Duration of the operation
- (4) Log entry

Tip: You can customize the logging format with `set_format_string()`, and set the maximum log entry length with the properties:

- `abbreviated_max_len_ascii`
- `abbreviated_max_len_bin`
- `abbreviated_max_len_list`

See the full logger help *here*.

Notice the SCPI communication starts from the line `driver.utilities.reset()`. If you want to log the initialization of the session as well, you have to switch the logging ON already in the constructor:

```
driver = RsCMPX_WlanMeas('TCPIP::192.168.56.101::hislip0', options='LoggingMode=On')
```

Parallel to the console logging, you can log to a general stream. Do not fear the programmer's jargon… under the term **stream** you can just imagine a file. To be a little more technical, a stream in Python is any object that has two methods: `write()` and `flush()`. This example opens a file and sets it as logging target:

```
"""
Example of logging to a file
"""

from RsCMPX_WlanMeas import *

driver = RsCMPX_WlanMeas('TCPIP::192.168.1.101::INSTR')
```

(continues on next page)

(continued from previous page)

```
# We also want to log to the console.
driver.utilities.logger.log_to_console = True

# Logging target is our file
file = open(r'c:\temp\my_file.txt', 'w')
driver.utilities.logger.set_logging_target(file)
driver.utilities.logger.mode = LoggingMode.On

# Instead of the 'TCPIP::192.168.1.101::INSTR', show 'MyDevice'
driver.utilities.logger.device_name = 'MyDevice'

# Custom user entry
driver.utilities.logger.info_raw('----- This is my custom log entry. ---- ')

driver.utilities.reset()

# Close the session
driver.close()

# Close the log file
file.close()
```

Tip: To make the log more compact, you can skip all the lines with `Status check: OK`:

```
driver.utilities.logger.log_status_check_ok = False
```

Hint: You can share the logging file between multiple sessions. In such case, remember to close the file only after you have stopped logging in all your sessions, otherwise you get a log write error.

For logging to a UDP port in addition to other log targets, use one of the lines:

```
driver.utilities.logger.log_to_udp = True
driver.utilities.logger.log_to_console_and_udp = True
```

You can select the UDP port to log to, the default is 49200:

```
driver.utilities.logger.udp_port = 49200
```

Another cool feature is logging only errors. To make this mode usefull for troubleshooting, you also want to see the circumstances which lead to the errors. Each driver elementary operation, for example, `write_str()`, can generate a group of log entries - let us call them **Segment**. In the logging mode `Errors`, a whole segment is logged only if at least one entry of the segment is an error.

The script below demonstrates this feature. We use a direct SCPI communication to send a misspelled SCPI command *CLS, which leads to instrument status error:

```
"""
Logging example to the console with only errors logged
"""
```

(continues on next page)

(continued from previous page)

```
from RsCMPX_WlanMeas import *

driver = RsCMPX_WlanMeas('TCPIP::192.168.1.101::INSTR', options='LoggingMode=Errors')

# Switch ON logging to the console.
driver.utilities.logger.log_to_console = True

# Reset will not be logged, since no error occurred there
driver.utilities.reset()

# Now a misspelled command.
driver.utilities.write('*CLaS')

# A good command again, no logging here
idn = driver.utilities.query('*IDN?')

# Close the session
driver.close()
```

Console output:

```
12:11:02.879 TCPIP::192.168.1.101::INSTR     0.976 ms  Write string: *CLaS
12:11:02.879 TCPIP::192.168.1.101::INSTR     6.833 ms  Status check: StatusException:
                                                       Instrument error detected: Undefined header;
↪*CLaS
```

Notice the following:

- Although the operation **Write string: *CLaS** finished without an error, it is still logged, because it provides the context for the actual error which occurred during the status checking right after.
- No other log entries are present, including the session initialization and close, because they were all error-free.

CHAPTER

THREE

ENUMS

3.1 Bandwidth

```
# Example value:
value = enums.Bandwidth.BW05mhz
# All values (8x):
BW05mhz | BW10mhz | BW16mhz | BW20mhz | BW32mhz | BW40mhz | BW80mhz | BW88mhz
```

3.2 BurstEvalLength

```
# Example value:
value = enums.BurstEvalLength.REDucedburst
# All values (2x):
REDucedburst | WHOLeburst
```

3.3 BurstType

```
# Example value:
value = enums.BurstType.AUTO
# All values (4x):
AUTO | DLIN | GREenfield | MIXed
```

3.4 BurstTypeB

```
# Example value:
value = enums.BurstTypeB.GREenfield
# All values (2x):
GREenfield | MIXed
```

3.5 CfoEstimation

```
# Example value:
value = enums.CfoEstimation.FULLpacket
# All values (2x):
FULLpacket | PREamble
```

3.6 ChannelEstimation

```
# Example value:
value = enums.ChannelEstimation.PAYLoad
# All values (2x):
PAYLoad | PREamble
```

3.7 Coderate

```
# Example value:
value = enums.Coderate.AUTO
# All values (7x):
AUTO | CR12 | CR14dcm | CR23 | CR34 | CR38dcm | CR56
```

3.8 CodingType

```
# Example value:
value = enums.CodingType.BCC
# All values (2x):
BCC | LDPC
```

3.9 ConnectorSwitch

```
# First value:
value = enums.ConnectorSwitch.R11
# Last value:
value = enums.ConnectorSwitch.RH8
# All values (96x):
R11 | R12 | R13 | R14 | R15 | R16 | R17 | R18
R21 | R22 | R23 | R24 | R25 | R26 | R27 | R28
R31 | R32 | R33 | R34 | R35 | R36 | R37 | R38
R41 | R42 | R43 | R44 | R45 | R46 | R47 | R48
RA1 | RA2 | RA3 | RA4 | RA5 | RA6 | RA7 | RA8
RB1 | RB2 | RB3 | RB4 | RB5 | RB6 | RB7 | RB8
RC1 | RC2 | RC3 | RC4 | RC5 | RC6 | RC7 | RC8
RD1 | RD2 | RD3 | RD4 | RD5 | RD6 | RD7 | RD8
```

(continues on next page)

(continued from previous page)

```
RE1 | RE2 | RE3 | RE4 | RE5 | RE6 | RE7 | RE8
RF1 | RF2 | RF3 | RF4 | RF5 | RF6 | RF7 | RF8
RG1 | RG2 | RG3 | RG4 | RG5 | RG6 | RG7 | RG8
RH1 | RH2 | RH3 | RH4 | RH5 | RH6 | RH7 | RH8
```

3.10 ConnectorSwitchExt

```
# First value:
value = enums.ConnectorSwitchExt.OFF
# Last value:
value = enums.ConnectorSwitchExt.RH8
# All values (98x):
OFF | ON | R11 | R12 | R13 | R14 | R15 | R16
R17 | R18 | R21 | R22 | R23 | R24 | R25 | R26
R27 | R28 | R31 | R32 | R33 | R34 | R35 | R36
R37 | R38 | R41 | R42 | R43 | R44 | R45 | R46
R47 | R48 | RA1 | RA2 | RA3 | RA4 | RA5 | RA6
RA7 | RA8 | RB1 | RB2 | RB3 | RB4 | RB5 | RB6
RB7 | RB8 | RC1 | RC2 | RC3 | RC4 | RC5 | RC6
RC7 | RC8 | RD1 | RD2 | RD3 | RD4 | RD5 | RD6
RD7 | RD8 | RE1 | RE2 | RE3 | RE4 | RE5 | RE6
RE7 | RE8 | RF1 | RF2 | RF3 | RF4 | RF5 | RF6
RF7 | RF8 | RG1 | RG2 | RG3 | RG4 | RG5 | RG6
RG7 | RG8 | RH1 | RH2 | RH3 | RH4 | RH5 | RH6
RH7 | RH8
```

3.11 ConnectorTuple

```
# Example value:
value = enums.ConnectorTuple.CT12
# All values (7x):
CT12 | CT14 | CT18 | CT34 | CT56 | CT58 | CT78
```

3.12 DecodeStatus

```
# Example value:
value = enums.DecodeStatus.INV
# All values (3x):
INV | NAV | OK
```

3.13 DisplayMode

```
# Example value:
value = enums.DisplayMode.ABSolute
# All values (2x):
ABSolute | RELative
```

3.14 EvmMethod

```
# Example value:
value = enums.EvmMethod.ST1999
# All values (3x):
ST1999 | ST2007 | ST2016
```

3.15 FftOffset

```
# Example value:
value = enums.FftOffset.AUTO
# All values (3x):
AUTO | CENT | PEAK
```

3.16 FrequencyBand

```
# Example value:
value = enums.FrequencyBand.B24Ghz
# All values (4x):
B24Ghz | B4GHz | B5GHz | B6GHz
```

3.17 GuardInterval

```
# Example value:
value = enums.GuardInterval.GI08
# All values (5x):
GI08 | GI16 | GI32 | LONG | SHORt
```

3.18 GuiScenario

```
# Example value:
value = enums.GuiScenario.CSPath
# All values (8x):
CSPath | MIMO2x2 | MIMO4x4 | MIMO8x8 | SALone | SMI4 | TMIMo | UNDefined
```

3.19 IeeeStandard

```
# Example value:
value = enums.IeeeStandard.DSSS
# All values (7x):
DSSS | EHTofdm | HEOFdm | HTOFdm | LOFDm | POFDm | VHTofdm
```

3.20 LowHigh

```
# Example value:
value = enums.LowHigh.HIGH
# All values (2x):
HIGH | LOW
```

3.21 LtfSize

```
# Example value:
value = enums.LtfSize.LTF1
# All values (3x):
LTF1 | LTF2 | LTF4
```

3.22 MimoScenario

```
# First value:
value = enums.MimoScenario.CSPath
# Last value:
value = enums.MimoScenario.UNDefined
# All values (10x):
CSPath | MIMO2x2 | MIMO4x4 | MIMO8x8 | SALone | SMI4 | TMIM2x2 | TMIM3x3
TMIM4x4 | UNDefined
```

3.23 ModulationFilter

```
# First value:
value = enums.ModulationFilter.ALL
# Last value:
value = enums.ModulationFilter.QPSK
# All values (11x):
ALL | BPSK | CCK11 | CCK5_5 | DBPSk | DQPSk | QAM1024 | QAM16
QAM256 | QAM64 | QPSK
```

3.24 ModulationTypeB

```
# First value:
value = enums.ModulationTypeB.BP1_5
# Last value:
value = enums.ModulationTypeB.QR34
# All values (32x):
BP1_5 | BP2_25 | BP3 | BP4_5 | BPM6 | BPM9 | BR12 | Q1M12
Q1M18 | Q1M24 | Q1M36 | Q1M6 | Q1M9 | Q1R12 | Q1R34 | Q6M12
Q6M135 | Q6M24 | Q6M27 | Q6M48 | Q6M54 | Q6R23 | Q6R34 | Q6R56
QM12 | QM18 | QM3 | QM4_5 | QM6 | QM9 | QR12 | QR34
```

3.25 ModulationTypeC

```
# Example value:
value = enums.ModulationTypeC.CCK11
# All values (4x):
CCK11 | CCK5 | DBPSk1 | DQPSk2
```

3.26 ModulationTypeD

```
# First value:
value = enums.ModulationTypeD._16Q
# Last value:
value = enums.ModulationTypeD.UNSPecified
# All values (28x):
_16Q | _16Q12 | _16Q14 | _16Q34 | _16Q38 | _1KQ | _1KQ34 | _1KQ56
_256Q | _256Q34 | _256Q56 | _4KQ | _4KQ34 | _4KQ56 | _64Q | _64Q12
_64Q23 | _64Q34 | _64Q56 | BPSK | BPSK12 | BPSK14 | BPSK34 | QPSK
QPSK12 | QPSK14 | QPSK34 | UNSPecified
```

3.27 ParameterSetMode

```
# Example value:
value = enums.ParameterSetMode.GLOBal
# All values (2x):
GLOBal | LIST
```

3.28 PlcpType

```
# Example value:
value = enums.PlcpType.LONGplcp
# All values (2x):
LONGplcp | SHORtplcp
```

3.29 PowerClass

```
# Example value:
value = enums.PowerClass.CLA
# All values (4x):
CLA | CLB | CLCD | USERdefined
```

3.30 ReceiveMode

```
# Example value:
value = enums.ReceiveMode.CMIMo
# All values (4x):
CMIMo | SISO | SMIMo | TMIMo
```

3.31 RefPower

```
# Example value:
value = enums.RefPower.MAXimum
# All values (2x):
MAXimum | MEAN
```

3.32 Repeat

```
# Example value:
value = enums.Repeat.CONTinuous
# All values (2x):
CONTinuous | SINGleshot
```

3.33 ResourceState

```
# Example value:
value = enums.ResourceState.ACTive
# All values (8x):
ACTive | ADJusted | INValid | OFF | PENDing | QUEued | RDY | RUN
```

3.34 ResultStatus2

```
# First value:
value = enums.ResultStatus2.DC
# Last value:
value = enums.ResultStatus2.ULEU
# All values (10x):
DC | INV | NAV | NCAP | OFF | OFL | OK | UFL
ULEL | ULEU
```

3.35 RxConnector

```
# First value:
value = enums.RxConnector.I11I
# Last value:
value = enums.RxConnector.RH8
# All values (219x):
I11I | I12O | I13I | I14O | I15I | I16O | I17I | I18O
I21I | I22O | I23I | I24O | I25I | I26O | I27I | I28O
I31I | I32O | I33I | I34O | I35I | I36O | I37I | I38O
I41I | I42O | I43I | I44O | I45I | I46O | I47I | I48O
IFI1 | IFI2 | IFI3 | IFI4 | IFI5 | IFI6 | IFO1 | IFO2
IFO3 | IFO4 | IFO5 | IFO6 | IQ1I | IQ2O | IQ3I | IQ4O
IQ5I | IQ6O | IQ7I | IQ8O | R10D | R11 | R118 | R1183
R1184 | R11C | R11D | R11O | R11O3 | R11O4 | R12 | R12C
R12D | R12I | R13 | R13C | R13O | R14 | R14C | R14I
R15 | R16 | R17 | R18 | R21 | R214 | R218 | R21C
R21O | R22 | R22C | R22I | R23 | R23C | R23O | R24
R24C | R24I | R25 | R258 | R26 | R27 | R28 | R31
R318 | R31C | R31O | R32 | R32C | R32I | R33 | R33C
R33O | R34 | R34C | R34I | R35 | R36 | R37 | R38
```

(continues on next page)

(continued from previous page)

```
R41 | R418 | R41C | R41O | R42 | R42C | R42I | R43
R43C | R43O | R44 | R44C | R44I | R45 | R46 | R47
R48 | RA1 | RA18 | RA2 | RA3 | RA4 | RA5 | RA6
RA7 | RA8 | RB1 | RB14 | RB18 | RB2 | RB3 | RB4
RB5 | RB6 | RB7 | RB8 | RC1 | RC18 | RC2 | RC3
RC4 | RC5 | RC6 | RC7 | RC8 | RD1 | RD18 | RD2
RD3 | RD4 | RD5 | RD6 | RD7 | RD8 | RE1 | RE18
RE2 | RE3 | RE4 | RE5 | RE6 | RE7 | RE8 | RF1
RF18 | RF1C | RF1O | RF2 | RF2C | RF2I | RF3 | RF3C
RF3O | RF4 | RF4C | RF4I | RF5 | RF5C | RF6 | RF6C
RF7 | RF7C | RF8 | RF8C | RF9C | RFAC | RFAO | RFBC
RFBI | RG1 | RG18 | RG2 | RG3 | RG4 | RG5 | RG6
RG7 | RG8 | RH1 | RH18 | RH2 | RH3 | RH4 | RH5
RH6 | RH7 | RH8
```

3.36 RxConnectorExt

```
# First value:
value = enums.RxConnectorExt.I11I
# Last value:
value = enums.RxConnectorExt.RH8
# All values (163x):
I11I | I13I | I15I | I17I | I21I | I23I | I25I | I27I
I31I | I33I | I35I | I37I | I41I | I43I | I45I | I47I
IFI1 | IFI2 | IFI3 | IFI4 | IFI5 | IFI6 | IQ1I | IQ3I
IQ5I | IQ7I | R10D | R11 | R11C | R11D | R12 | R12C
R12D | R12I | R13 | R13C | R14 | R14C | R14I | R15
R16 | R17 | R18 | R21 | R21C | R22 | R22C | R22I
R23 | R23C | R24 | R24C | R24I | R25 | R26 | R27
R28 | R31 | R31C | R32 | R32C | R32I | R33 | R33C
R34 | R34C | R34I | R35 | R36 | R37 | R38 | R41
R41C | R42 | R42C | R42I | R43 | R43C | R44 | R44C
R44I | R45 | R46 | R47 | R48 | RA1 | RA2 | RA3
RA4 | RA5 | RA6 | RA7 | RA8 | RB1 | RB2 | RB3
RB4 | RB5 | RB6 | RB7 | RB8 | RC1 | RC2 | RC3
RC4 | RC5 | RC6 | RC7 | RC8 | RD1 | RD2 | RD3
RD4 | RD5 | RD6 | RD7 | RD8 | RE1 | RE2 | RE3
RE4 | RE5 | RE6 | RE7 | RE8 | RF1 | RF1C | RF2
RF2C | RF2I | RF3 | RF3C | RF4 | RF4C | RF4I | RF5
RF5C | RF6 | RF6C | RF7 | RF7C | RF8 | RF8C | RF9C
RFAC | RFBC | RFBI | RG1 | RG2 | RG3 | RG4 | RG5
RG6 | RG7 | RG8 | RH1 | RH2 | RH3 | RH4 | RH5
RH6 | RH7 | RH8
```

3.37 RxConverter

```
# First value:
value = enums.RxConverter.IRX1
# Last value:
value = enums.RxConverter.RX44
# All values (40x):
IRX1 | IRX11 | IRX12 | IRX13 | IRX14 | IRX2 | IRX21 | IRX22
IRX23 | IRX24 | IRX3 | IRX31 | IRX32 | IRX33 | IRX34 | IRX4
IRX41 | IRX42 | IRX43 | IRX44 | RX1 | RX11 | RX12 | RX13
RX14 | RX2 | RX21 | RX22 | RX23 | RX24 | RX3 | RX31
RX32 | RX33 | RX34 | RX4 | RX41 | RX42 | RX43 | RX44
```

3.38 RxTxConverter

```
# First value:
value = enums.RxTxConverter.IRX1
# Last value:
value = enums.RxTxConverter.TX44
# All values (80x):
IRX1 | IRX11 | IRX12 | IRX13 | IRX14 | IRX2 | IRX21 | IRX22
IRX23 | IRX24 | IRX3 | IRX31 | IRX32 | IRX33 | IRX34 | IRX4
IRX41 | IRX42 | IRX43 | IRX44 | ITX1 | ITX11 | ITX12 | ITX13
ITX14 | ITX2 | ITX21 | ITX22 | ITX23 | ITX24 | ITX3 | ITX31
ITX32 | ITX33 | ITX34 | ITX4 | ITX41 | ITX42 | ITX43 | ITX44
RX1 | RX11 | RX12 | RX13 | RX14 | RX2 | RX21 | RX22
RX23 | RX24 | RX3 | RX31 | RX32 | RX33 | RX34 | RX4
RX41 | RX42 | RX43 | RX44 | TX1 | TX11 | TX12 | TX13
TX14 | TX2 | TX21 | TX22 | TX23 | TX24 | TX3 | TX31
TX32 | TX33 | TX34 | TX4 | TX41 | TX42 | TX43 | TX44
```

3.39 SlopeType

```
# Example value:
value = enums.SlopeType.NEGative
# All values (2x):
NEGative | POSitive
```

3.40 StopCondition

```
# Example value:
value = enums.StopCondition.NONE
# All values (2x):
NONE | SLFail
```

3.41 SynchroMode

```
# Example value:
value = enums.SynchroMode.NORMal
# All values (2x):
NORMal | TOLerant
```

3.42 TrainingMode

```
# Example value:
value = enums.TrainingMode.MMODe
# All values (2x):
MMODe | TMODe
```

3.43 TriggerSlope

```
# Example value:
value = enums.TriggerSlope.FEDGe
# All values (4x):
FEDGe | OFF | ON | REDGe
```

CHAPTER

FOUR

REPCAPS

4.1 Instance (Global)

```
# Setting:
driver.repcap_instance_set(repcap.Instance.Inst1)
# Range:
Inst1 .. Inst32
# All values (32x):
Inst1 | Inst2 | Inst3 | Inst4 | Inst5 | Inst6 | Inst7 | Inst8
Inst9 | Inst10 | Inst11 | Inst12 | Inst13 | Inst14 | Inst15 | Inst16
Inst17 | Inst18 | Inst19 | Inst20 | Inst21 | Inst22 | Inst23 | Inst24
Inst25 | Inst26 | Inst27 | Inst28 | Inst29 | Inst30 | Inst31 | Inst32
```

4.2 Antenna

```
# First value:
value = repcap.Antenna.Nr1
# Range:
Nr1 .. Nr8
# All values (8x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
```

4.3 Band

```
# First value:
value = repcap.Band.Nr2
# Values (2x):
Nr2 | Nr5
```

4.4 BandwidthA

```
# First value:
value = repcap.BandwidthA.Bw10
# Values (2x):
Bw10 | Bw20
```

4.5 BandwidthB

```
# First value:
value = repcap.BandwidthB.Bw5
# Values (3x):
Bw5 | Bw10 | Bw20
```

4.6 BandwidthC

```
# First value:
value = repcap.BandwidthC.Bw5
# Values (4x):
Bw5 | Bw10 | Bw20 | Bw40
```

4.7 BandwidthD

```
# First value:
value = repcap.BandwidthD.Bw20
# Range:
Bw20 .. Bw8080
# All values (5x):
Bw20 | Bw40 | Bw80 | Bw160 | Bw8080
```

4.8 BandwidthE

```
# First value:
value = repcap.BandwidthE.Bw5
# Range:
Bw5 .. Bw8080
# All values (7x):
Bw5 | Bw10 | Bw20 | Bw40 | Bw80 | Bw160 | Bw8080
```

4.9 BandwidthF

```
# First value:
value = repcap.BandwidthF.Bw20
# Range:
Bw20 .. Bw320
# All values (5x):
Bw20 | Bw40 | Bw80 | Bw160 | Bw320
```

4.10 BandwidthG

```
# First value:
value = repcap.BandwidthG.Bw5
# Range:
Bw5 .. Bw320
# All values (7x):
Bw5 | Bw10 | Bw20 | Bw40 | Bw80 | Bw160 | Bw320
```

4.11 Channel

```
# First value:
value = repcap.Channel.Nr1
# Values (2x):
Nr1 | Nr2
```

4.12 Channels

```
# First value:
value = repcap.Channels.Nr1
# Range:
Nr1 .. Nr8
# All values (8x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
```

4.13 Connector

```
# First value:
value = repcap.Connector.Nr1
# Values (4x):
Nr1 | Nr2 | Nr3 | Nr4
```

4.14 Mimo

```
# First value:
value = repcap.Mimo.Nr1
# Range:
Nr1 .. Nr8
# All values (8x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
```

4.15 Reserved

```
# First value:
value = repcap.Reserved.Nr1
# Values (3x):
Nr1 | Nr2 | Nr3
```

4.16 ResourceUnit

```
# First value:
value = repcap.ResourceUnit.Nr1
# Range:
Nr1 .. Nr144
# All values (144x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
Nr9 | Nr10 | Nr11 | Nr12 | Nr13 | Nr14 | Nr15 | Nr16
Nr17 | Nr18 | Nr19 | Nr20 | Nr21 | Nr22 | Nr23 | Nr24
Nr25 | Nr26 | Nr27 | Nr28 | Nr29 | Nr30 | Nr31 | Nr32
Nr33 | Nr34 | Nr35 | Nr36 | Nr37 | Nr38 | Nr39 | Nr40
Nr41 | Nr42 | Nr43 | Nr44 | Nr45 | Nr46 | Nr47 | Nr48
Nr49 | Nr50 | Nr51 | Nr52 | Nr53 | Nr54 | Nr55 | Nr56
Nr57 | Nr58 | Nr59 | Nr60 | Nr61 | Nr62 | Nr63 | Nr64
Nr65 | Nr66 | Nr67 | Nr68 | Nr69 | Nr70 | Nr71 | Nr72
Nr73 | Nr74 | Nr75 | Nr76 | Nr77 | Nr78 | Nr79 | Nr80
Nr81 | Nr82 | Nr83 | Nr84 | Nr85 | Nr86 | Nr87 | Nr88
Nr89 | Nr90 | Nr91 | Nr92 | Nr93 | Nr94 | Nr95 | Nr96
Nr97 | Nr98 | Nr99 | Nr100 | Nr101 | Nr102 | Nr103 | Nr104
Nr105 | Nr106 | Nr107 | Nr108 | Nr109 | Nr110 | Nr111 | Nr112
Nr113 | Nr114 | Nr115 | Nr116 | Nr117 | Nr118 | Nr119 | Nr120
Nr121 | Nr122 | Nr123 | Nr124 | Nr125 | Nr126 | Nr127 | Nr128
Nr129 | Nr130 | Nr131 | Nr132 | Nr133 | Nr134 | Nr135 | Nr136
Nr137 | Nr138 | Nr139 | Nr140 | Nr141 | Nr142 | Nr143 | Nr144
```

4.17 RxAntenna

```
# First value:
value = repcap.RxAntenna.Nr1
# Range:
Nr1 .. Nr8
# All values (8x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
```

4.18 Segment

```
# First value:
value = repcap.Segment.Nr1
# Values (2x):
Nr1 | Nr2
```

4.19 SegmentB

```
# First value:
value = repcap.SegmentB.Nr1
# Range:
Nr1 .. Nr32
# All values (32x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
Nr9 | Nr10 | Nr11 | Nr12 | Nr13 | Nr14 | Nr15 | Nr16
Nr17 | Nr18 | Nr19 | Nr20 | Nr21 | Nr22 | Nr23 | Nr24
Nr25 | Nr26 | Nr27 | Nr28 | Nr29 | Nr30 | Nr31 | Nr32
```

4.20 Smi

```
# First value:
value = repcap.Smi.Nr4
# Values (1x):
Nr4
```

4.21 SMimoPath

```
# First value:
value = repcap.SMimoPath.Count2
# Values (3x):
Count2 | Count4 | Count8
```

4.22 Spatial

```
# First value:
value = repcap.Spatial.Nr1
# Values (4x):
Nr1 | Nr2 | Nr3 | Nr4
```

4.23 Stream

```
# First value:
value = repcap.Stream.Nr1
# Range:
Nr1 .. Nr8
# All values (8x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
```

4.24 TrueMimoPath

```
# First value:
value = repcap.TrueMimoPath.Count1
# Values (4x):
Count1 | Count2 | Count3 | Count4
```

4.25 User

```
# First value:
value = repcap.User.Nr1
# Range:
Nr1 .. Nr144
# All values (144x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
Nr9 | Nr10 | Nr11 | Nr12 | Nr13 | Nr14 | Nr15 | Nr16
Nr17 | Nr18 | Nr19 | Nr20 | Nr21 | Nr22 | Nr23 | Nr24
Nr25 | Nr26 | Nr27 | Nr28 | Nr29 | Nr30 | Nr31 | Nr32
Nr33 | Nr34 | Nr35 | Nr36 | Nr37 | Nr38 | Nr39 | Nr40
Nr41 | Nr42 | Nr43 | Nr44 | Nr45 | Nr46 | Nr47 | Nr48
Nr49 | Nr50 | Nr51 | Nr52 | Nr53 | Nr54 | Nr55 | Nr56
Nr57 | Nr58 | Nr59 | Nr60 | Nr61 | Nr62 | Nr63 | Nr64
Nr65 | Nr66 | Nr67 | Nr68 | Nr69 | Nr70 | Nr71 | Nr72
Nr73 | Nr74 | Nr75 | Nr76 | Nr77 | Nr78 | Nr79 | Nr80
Nr81 | Nr82 | Nr83 | Nr84 | Nr85 | Nr86 | Nr87 | Nr88
Nr89 | Nr90 | Nr91 | Nr92 | Nr93 | Nr94 | Nr95 | Nr96
Nr97 | Nr98 | Nr99 | Nr100 | Nr101 | Nr102 | Nr103 | Nr104
Nr105 | Nr106 | Nr107 | Nr108 | Nr109 | Nr110 | Nr111 | Nr112
Nr113 | Nr114 | Nr115 | Nr116 | Nr117 | Nr118 | Nr119 | Nr120
```

(continues on next page)

(continued from previous page)

```
Nr121 | Nr122 | Nr123 | Nr124 | Nr125 | Nr126 | Nr127 | Nr128
Nr129 | Nr130 | Nr131 | Nr132 | Nr133 | Nr134 | Nr135 | Nr136
Nr137 | Nr138 | Nr139 | Nr140 | Nr141 | Nr142 | Nr143 | Nr144
```

4.26 UserIx

```
# First value:
value = repcap.UserIx.Nr1
# Range:
Nr1 .. Nr144
# All values (144x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
Nr9 | Nr10 | Nr11 | Nr12 | Nr13 | Nr14 | Nr15 | Nr16
Nr17 | Nr18 | Nr19 | Nr20 | Nr21 | Nr22 | Nr23 | Nr24
Nr25 | Nr26 | Nr27 | Nr28 | Nr29 | Nr30 | Nr31 | Nr32
Nr33 | Nr34 | Nr35 | Nr36 | Nr37 | Nr38 | Nr39 | Nr40
Nr41 | Nr42 | Nr43 | Nr44 | Nr45 | Nr46 | Nr47 | Nr48
Nr49 | Nr50 | Nr51 | Nr52 | Nr53 | Nr54 | Nr55 | Nr56
Nr57 | Nr58 | Nr59 | Nr60 | Nr61 | Nr62 | Nr63 | Nr64
Nr65 | Nr66 | Nr67 | Nr68 | Nr69 | Nr70 | Nr71 | Nr72
Nr73 | Nr74 | Nr75 | Nr76 | Nr77 | Nr78 | Nr79 | Nr80
Nr81 | Nr82 | Nr83 | Nr84 | Nr85 | Nr86 | Nr87 | Nr88
Nr89 | Nr90 | Nr91 | Nr92 | Nr93 | Nr94 | Nr95 | Nr96
Nr97 | Nr98 | Nr99 | Nr100 | Nr101 | Nr102 | Nr103 | Nr104
Nr105 | Nr106 | Nr107 | Nr108 | Nr109 | Nr110 | Nr111 | Nr112
Nr113 | Nr114 | Nr115 | Nr116 | Nr117 | Nr118 | Nr119 | Nr120
Nr121 | Nr122 | Nr123 | Nr124 | Nr125 | Nr126 | Nr127 | Nr128
Nr129 | Nr130 | Nr131 | Nr132 | Nr133 | Nr134 | Nr135 | Nr136
Nr137 | Nr138 | Nr139 | Nr140 | Nr141 | Nr142 | Nr143 | Nr144
```

4.27 UtError

```
# First value:
value = repcap.UtError.Nr1
# Range:
Nr1 .. Nr8
# All values (8x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
```

CHAPTER

FIVE

EXAMPLES

For more examples, visit our Rohde & Schwarz Github repository.

```
""" Example on how to use the python RsCmw auto-generated instrument driver showing:
- usage of basic properties of the cmw_base object
- basic concept of setting commands and repcaps: DISPlay:WINDow<n>:SELect
- cmw_xxx drivers reliability interface usage
"""

from RsCmwBase import *  # install from pypi.org

RsCmwBase.assert_minimum_version('3.7.90.38')
cmw_base = RsCmwBase('TCPIP::10.112.1.116::INSTR', True, False)
print(f'CMW Base IND: {cmw_base.utilities.idn_string}')
print(f'CMW Instrument options:\n{",".join(cmw_base.utilities.instrument_options)}')
cmw_base.utilities.visa_timeout = 5000

# Sends OPC after each command
cmw_base.utilities.opc_query_after_write = False

# Checks for syst:err? after each command / query
cmw_base.utilities.instrument_status_checking = True

# DISPlay:WINDow<n>:SELect
cmw_base.display.window.select.set(repcap.Window.Win1)
cmw_base.display.window.repcap_window_set(repcap.Window.Win2)
cmw_base.display.window.select.set()

# Self-test
self_test = cmw_base.utilities.self_test()
print(f'CMW self-test result: {self_test} - {"Passed" if self_test[0] == 0 else "Failed"}
↪"')

# Driver's Interface reliability offers a convenient way of reacting on the return value␣
↪Reliability Indicator
cmw_base.reliability.ExceptionOnError = True


# Callback to use for the reliability indicator update event
def my_reliability_handler(event_args: ReliabilityEventArgs):
        print(f'Base Reliability updated.\nContext: {event_args.context}\nMessage:
```

(continues on next page)

(continued from previous page)

```
↪{event_args.message}')


# We register a callback for each change in the reliability indicator
cmw_base.reliability.on_update_handler = my_reliability_handler

# You can obtain the last value of the returned reliability
print(f"\nReliability last value: {cmw_base.reliability.last_value}, context '{cmw_base.
↪reliability.last_context}', message: {cmw_base.reliability.last_message}")

# Reference Frequency Source
cmw_base.system.reference.frequency.set_source(enums.SourceIntExt.INTernal)

# Close the session
cmw_base.close()
```

CHAPTER

SIX

RSCMPX_WLANMEAS API STRUCTURE

Global RepCaps

```
driver = RsCMPX_WlanMeas('TCPIP::192.168.2.101::hislip0')
# Instance range: Inst1 .. Inst32
rc = driver.repcap_instance_get()
driver.repcap_instance_set(repcap.Instance.Inst1)
```

class RsCMPX_WlanMeas(*resource_name: str*, *id_query: bool = True*, *reset: bool = False*, *options: str = None*, *direct_session: object = None*)

1254 total commands, 4 Subgroups, 0 group commands

Initializes new RsCMPX_WlanMeas session.

Parameter options tokens examples:

- `Simulate=True` - starts the session in simulation mode. Default: `False`
- `SelectVisa=socket` - uses no VISA implementation for socket connections - you do not need any VISA-C installation
- `SelectVisa=rs` - forces usage of RohdeSchwarz Visa
- `SelectVisa=ivi` - forces usage of National Instruments Visa
- `QueryInstrumentStatus = False` - same as `driver.utilities.instrument_status_checking = False`. Default: `True`
- `WriteDelay = 20, ReadDelay = 5` - Introduces delay of 20ms before each write and 5ms before each read. Default: `0ms` for both
- `OpcWaitMode = OpcQuery` - mode for all the opc-synchronised write/reads. Other modes: StbPolling, StbPollingSlow, StbPollingSuperSlow. Default: `StbPolling`
- `AddTermCharToWriteBinBLock = True` - Adds one additional LF to the end of the binary data (some instruments require that). Default: `False`
- `AssureWriteWithTermChar = True` - Makes sure each command/query is terminated with termination character. Default: Interface dependent
- `TerminationCharacter = "\r"` - Sets the termination character for reading. Default: `\n` (LineFeed or LF)
- `DataChunkSize = 10E3` - Maximum size of one write/read segment. If transferred data is bigger, it is split to more segments. Default: `1E6` bytes
- `OpcTimeout = 10000` - same as driver.utilities.opc_timeout = 10000. Default: `30000ms`
- `VisaTimeout = 5000` - same as driver.utilities.visa_timeout = 5000. Default: `10000ms`

- `ViClearExeMode = Disabled` - viClear() execution mode. Default: `execute_on_all`
- `OpcQueryAfterWrite = True` - same as driver.utilities.opc_query_after_write = True. Default: `False`
- `StbInErrorCheck = False` - if true, the driver checks errors with *STB? If false, it uses SYST:ERR?. Default: `True`
- `ScpiQuotes = double'. - for SCPI commands, you can define how strings are quoted. With single or double quotes. Possible values: single | double | {char}. Default: ``single`
- `LoggingMode = On` - Sets the logging status right from the start. Default: `Off`
- `LoggingName = 'MyDevice'` - Sets the name to represent the session in the log entries. Default: `'resource_name'`
- `LogToGlobalTarget = True` - Sets the logging target to the class-property previously set with RsCMPX_WlanMeas.set_global_logging_target() Default: `False`
- `LoggingToConsole = True` - Immediately starts logging to the console. Default: False
- `LoggingToUdp = True` - Immediately starts logging to the UDP port. Default: False
- `LoggingUdpPort = 49200` - UDP port to log to. Default: 49200

Parameters

- **resource_name** – VISA resource name, e.g. 'TCPIP::192.168.2.1::INSTR'
- **id_query** – if True, the instrument's model name is verified against the models supported by the driver and eventually throws an exception.
- **reset** – Resets the instrument (sends *RST command) and clears its status sybsystem.
- **options** – string tokens alternating the driver settings.
- **direct_session** – Another driver object or pyVisa object to reuse the session instead of opening a new session.

static assert_minimum_version(*min_version: str*) → None

Asserts that the driver version fulfills the minimum required version you have entered. This way you make sure your installed driver is of the entered version or newer.

classmethod clear_global_logging_relative_timestamp() → None

Clears the global relative timestamp. After this, all the instances using the global relative timestamp continue logging with the absolute timestamps.

close() → None

Closes the active RsCMPX_WlanMeas session.

classmethod from_existing_session(*session: object*, *options: str = None*) → RsCMPX_WlanMeas

Creates a new RsCMPX_WlanMeas object with the entered 'session' reused.

Parameters

- **session** – can be another driver or a direct pyvisa session.
- **options** – string tokens alternating the driver settings.

classmethod get_global_logging_relative_timestamp() → datetime

Returns global common relative timestamp for log entries.

classmethod get_global_logging_target()

Returns global common target stream.

get_session_handle() → object

Returns the underlying session handle.

get_total_execution_time() → timedelta

Returns total time spent by the library on communicating with the instrument. This time is always shorter than get_total_time(), since it does not include gaps between the communication. You can reset this counter with reset_time_statistics().

get_total_time() → timedelta

Returns total time spent by the library on communicating with the instrument. This time is always shorter than get_total_time(), since it does not include gaps between the communication. You can reset this counter with reset_time_statistics().

static list_resources(*expression: str = '?*::INSTR'*, *visa_select: str = None*) → List[str]

Finds all the resources defined by the expression

- '?*' - matches all the available instruments
- 'USB::?*' - matches all the USB instruments
- 'TCPIP::192?*' - matches all the LAN instruments with the IP address starting with 192

Parameters

- **expression** – see the examples in the function
- **visa_select** – optional parameter selecting a specific VISA. Examples: '@ivi', '@rs'

reset_time_statistics() → None

Resets all execution and total time counters. Affects the results of get_total_time() and get_total_execution_time()

restore_all_repcaps_to_default() → None

Sets all the Group and Global repcaps to their initial values

classmethod set_global_logging_relative_timestamp(*timestamp: datetime*) → None

Sets global common relative timestamp for log entries. To use it, call the following: io.utilities.logger.set_relative_timestamp_global()

classmethod set_global_logging_relative_timestamp_now() → None

Sets global common relative timestamp for log entries to this moment. To use it, call the following: io.utilities.logger.set_relative_timestamp_global().

classmethod set_global_logging_target(*target*) → None

Sets global common target stream that each instance can use. To use it, call the following: io.utilities.logger.set_logging_target_global(). If an instance uses global logging target, it automatically uses the global relative timestamp (if set). You can set the target to None to invalidate it.

Subgroups

6.1 Configure

class ConfigureCls

Configure commands group definition. 222 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.clone()
```

Subgroups

6.1.1 WlanMeas

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MODE
```

class WlanMeasCls

WlanMeas commands group definition. 222 total commands, 6 Subgroups, 1 group commands

get_mode() → TrainingMode

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MODE
value: enums.TrainingMode = driver.configure.wlanMeas.get_mode()
```

No command help available

return
training_mode: No help available

set_mode(*training_mode: TrainingMode*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MODE
driver.configure.wlanMeas.set_mode(training_mode = enums.TrainingMode.MMODe)
```

No command help available

param training_mode
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.clone()
```

Subgroups

6.1.1.1 Isignal

SCPI Commands :

```
CONFigure:WLAN:MEASurement<Instance>:ISIGnal:STANdard
CONFigure:WLAN:MEASurement<Instance>:ISIGnal:RMODe
CONFigure:WLAN:MEASurement<Instance>:ISIGnal:ELENgth
CONFigure:WLAN:MEASurement<Instance>:ISIGnal:BTYPe
CONFigure:WLAN:MEASurement<Instance>:ISIGnal:BWIDth
CONFigure:WLAN:MEASurement<Instance>:ISIGnal:CDIStance
CONFigure:WLAN:MEASurement<Instance>:ISIGnal:PCLass
CONFigure:WLAN:MEASurement<Instance>:ISIGnal:IQSWap
CONFigure:WLAN:MEASurement<Instance>:ISIGnal:MODFilter
```

class IsignalCls

Isignal commands group definition. 13 total commands, 3 Subgroups, 9 group commands

get_bandwidth() → Bandwidth

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:ISIGnal:BWIDth
value: enums.Bandwidth = driver.configure.wlanMeas.isignal.get_bandwidth()
```

Selects the channel bandwidth.

return
bandwidth: BW05mhz: 5 MHz BW10mhz: 10 MHz BW20mhz: 20 MHz BW40mhz: 40 MHz BW80mhz: 80 MHz BW16mhz: 160 MHz BW32mhz: 320 MHz

get_btype() → BurstType

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:ISIGnal:BTYPe
value: enums.BurstType = driver.configure.wlanMeas.isignal.get_btype()
```

Sets the burst type for standard 802.11n. Do not use the command for other standards.

return
burst_type: MIXed: Compatibility mode, for coexistence with older standards GREenfield: Greenfield mode, incompatible with older standards

get_cdistance() → int

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:ISIGnal:CDIStance
value: int = driver.configure.wlanMeas.isignal.get_cdistance()
```

No command help available

return
channel_distance: No help available

get_elength() → BurstEvalLength

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:ISIGnal:ELENgth
value: enums.BurstEvalLength = driver.configure.wlanMeas.isignal.get_elength()
```

No command help available

return
evaluation_length: No help available

get_iqswap() → bool

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:ISIGnal:IQSWap
value: bool = driver.configure.wlanMeas.isignal.get_iqswap()
```

Swaps the role of the I and Q axes in the baseband.

return
iqswap: No help available

get_modfilter() → ModulationFilter

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:ISIGnal:MODFilter
value: enums.ModulationFilter = driver.configure.wlanMeas.isignal.get_
↪modfilter()
```

This command allows you to limit the evaluation to bursts of a particular modulation format. If the received burst has a different modulation, the reliability Wrong Modulation is displayed.

return
modulation_filter: Valid for OFDM: all, BPSK, QPSK, 16QAM, 64QAM6, 256QAM, 1024QAM, 4096QAM Valid for DSSS: all, DBPSK (1 Mbit/s) , DQPSK (2 Mbit/s) , CCK (5.5 Mbit/s) , CCK (11 Mbit/s)

get_pclass() → PowerClass

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:ISIGnal:PCLass
value: enums.PowerClass = driver.configure.wlanMeas.isignal.get_pclass()
```

Sets the STA transmit power class for 802.11p and selects the transmit spectrum mask to be applied.

return
power_class: CLA: class A transmit spectrum mask CLB: class B transmit spectrum mask CLCD: class C or D, no transmit spectrum limit check USERdefined: user-defined transmit spectrum mask

get_rmode() → ReceiveMode

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:ISIGnal:RMODe
value: enums.ReceiveMode = driver.configure.wlanMeas.isignal.get_rmode()
```

Sets the receive mode. Not all standards support MIMO. If you set a standard that is incompatible with the current receive mode, the receive mode automatically reverts to SISO.

return
receive_mode: SISO: SISO signal CMIMo: Composite MIMO TMIMo: True MIMO

get_standard() → IeeeStandard

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:ISIGnal:STANdard
value: enums.IeeeStandard = driver.configure.wlanMeas.isignal.get_standard()
```

Selects the IEEE 802.11 standard. Several WLAN signal properties depend on the selected standard, see 'Physical layer'.

return
standard: DSSS: 802.11b/g (DSSS) LOFDm: 802.11a/g (OFDM) HTOFdm: 802.11n VHTofdm: 802.11ac HEOFdm: 802.11ax POFDm: 802.11p EHTofdm: 802.11be

set_bandwidth(*bandwidth: Bandwidth*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:ISIGnal:BWIDth
driver.configure.wlanMeas.isignal.set_bandwidth(bandwidth = enums.Bandwidth.
↪BW05mhz)
```

Selects the channel bandwidth.

param bandwidth
BW05mhz: 5 MHz BW10mhz: 10 MHz BW20mhz: 20 MHz BW40mhz: 40 MHz BW80mhz: 80 MHz BW16mhz: 160 MHz BW32mhz: 320 MHz

set_btype(*burst_type: BurstType*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:ISIGnal:BTYPe
driver.configure.wlanMeas.isignal.set_btype(burst_type = enums.BurstType.AUTO)
```

Sets the burst type for standard 802.11n. Do not use the command for other standards.

param burst_type
MIXed: Compatibility mode, for coexistence with older standards GREenfield: Greenfield mode, incompatible with older standards

set_cdistance(*channel_distance: int*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:ISIGnal:CDIStance
driver.configure.wlanMeas.isignal.set_cdistance(channel_distance = 1)
```

No command help available

param channel_distance
No help available

set_elength(*evaluation_length: BurstEvalLength*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:ISIGnal:ELENgth
driver.configure.wlanMeas.isignal.set_elength(evaluation_length = enums.
↪BurstEvalLength.REDucedburst)
```

No command help available

param evaluation_length
No help available

set_iqswap(*iqswap: bool*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:ISIGnal:IQSWap
driver.configure.wlanMeas.isignal.set_iqswap(iqswap = False)
```

Swaps the role of the I and Q axes in the baseband.

param iqswap
No help available

set_modfilter(*modulation_filter: ModulationFilter*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:ISIGnal:MODFilter
driver.configure.wlanMeas.isignal.set_modfilter(modulation_filter = enums.
↪ModulationFilter.ALL)
```

This command allows you to limit the evaluation to bursts of a particular modulation format. If the received burst has a different modulation, the reliability Wrong Modulation is displayed.

param modulation_filter
Valid for OFDM: all, BPSK, QPSK, 16QAM, 64QAM6, 256QAM, 1024QAM, 4096QAM Valid for DSSS: all, DBPSK (1 Mbit/s) , DQPSK (2 Mbit/s) , CCK (5.5 Mbit/s) , CCK (11 Mbit/s)

set_pclass(*power_class: PowerClass*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:ISIGnal:PCLass
driver.configure.wlanMeas.isignal.set_pclass(power_class = enums.PowerClass.CLA)
```

Sets the STA transmit power class for 802.11p and selects the transmit spectrum mask to be applied.

param power_class
CLA: class A transmit spectrum mask CLB: class B transmit spectrum mask CLCD: class C or D, no transmit spectrum limit check USERdefined: user-defined transmit spectrum mask

set_rmode(*receive_mode: ReceiveMode*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:ISIGnal:RMODe
driver.configure.wlanMeas.isignal.set_rmode(receive_mode = enums.ReceiveMode.
↪CMIMo)
```

Sets the receive mode. Not all standards support MIMO. If you set a standard that is incompatible with the current receive mode, the receive mode automatically reverts to SISO.

param receive_mode
SISO: SISO signal CMIMo: Composite MIMO TMIMo: True MIMO

set_standard(*standard: IeeeStandard*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:ISIGnal:STANdard
driver.configure.wlanMeas.isignal.set_standard(standard = enums.IeeeStandard.
↪DSSS)
```

Selects the IEEE 802.11 standard. Several WLAN signal properties depend on the selected standard, see 'Physical layer'.

param standard
DSSS: 802.11b/g (DSSS) LOFDm: 802.11a/g (OFDM) HTOFdm: 802.11n VHTofdm: 802.11ac HEOFdm: 802.11ax POFDm: 802.11p EHTofdm: 802.11be

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.isignal.clone()
```

Subgroups

6.1.1.1.1 Dsss

class DsssCls

Dsss commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.isignal.dsss.clone()
```

Subgroups

6.1.1.1.1.1 Elength

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:ISIGnal:DSSS:ELENgth
```

class ElengthCls

Elength commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class ElengthStruct

Response structure. Fields:

- Evaluation_Length_Chips: int: No parameter help available
- Skip_Ph: bool: OFF: measure also preamble and header ON: skip preamble and header

get() → ElengthStruct

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:ISIGnal:DSSS:ELENgth
value: ElengthStruct = driver.configure.wlanMeas.isignal.dsss.elength.get()
```

Specifies the evaluation length of the burst for DSSS signals.

return
structure: for return value, see the help for ElengthStruct structure arguments.

set(*evaluation_length_chips: int*, *skip_ph: bool = None*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:ISIGnal:DSSS:ELENgth
driver.configure.wlanMeas.isignal.dsss.elength.set(evaluation_length_chips = 1,␣
↪skip_ph = False)
```

Specifies the evaluation length of the burst for DSSS signals.

param evaluation_length_chips
Number of payload chips

param skip_ph
OFF: measure also preamble and header ON: skip preamble and header

6.1.1.1.2 Ofdm

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:ISIGnal:OFDM:ELENgth
```

class OfdmCls

Ofdm commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_elength() → int

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:ISIGnal:OFDM:ELENgth
value: int = driver.configure.wlanMeas.isignal.ofdm.get_elength()
```

Specifies the evaluation length of the burst for OFDM signals.

return
evaluation_length_symbols: No help available

set_elength(*evaluation_length_symbols: int*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:ISIGnal:OFDM:ELENgth
driver.configure.wlanMeas.isignal.ofdm.set_elength(evaluation_length_symbols =␣
↪1)
```

Specifies the evaluation length of the burst for OFDM signals.

param evaluation_length_symbols
Number of payload symbols

6.1.1.1.3 Tdata

SCPI Command :

```
CONFigure:WLAN:MEASurement<instance>:ISIGnal:TDATa
```

class TdataCls

Tdata commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get_value() → str

```
# SCPI: CONFigure:WLAN:MEASurement<instance>:ISIGnal:TDATa
value: str = driver.configure.wlanMeas.isignal.tdata.get_value()
```

No command help available

return
filename: No help available

set_value(*filename: str*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<instance>:ISIGnal:TDATa
driver.configure.wlanMeas.isignal.tdata.set_value(filename = 'abc')
```

No command help available

param filename
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.isignal.tdata.clone()
```

Subgroups

6.1.1.1.3.1 File

SCPI Command :

```
CONFigure:WLAN:MEASurement<instance>:ISIGnal:TDATa:FILE:DATE
```

class FileCls

File commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_date() → str

```
# SCPI: CONFigure:WLAN:MEASurement<instance>:ISIGnal:TDATa:FILE:DATE
value: str = driver.configure.wlanMeas.isignal.tdata.file.get_date()
```

No command help available

return
file_date: No help available

6.1.1.2 Mimo

SCPI Command :

```
CONFigure:WLAN:MEASurement<instance>:MIMO:NOANtennas
```

class MimoCls

Mimo commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_no_antennas() → int

```
# SCPI: CONFigure:WLAN:MEASurement<instance>:MIMO:NOANtennas
value: int = driver.configure.wlanMeas.mimo.get_no_antennas()
```

Sets the number of connected antennas for SISO and MIMO measurements.

return
num_of_antennas: Number of antennas (1..4) , depending on receive mode.

set_no_antennas(*num_of_antennas: int*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<instance>:MIMO:NOANtennas
driver.configure.wlanMeas.mimo.set_no_antennas(num_of_antennas = 1)
```

Sets the number of connected antennas for SISO and MIMO measurements.

param num_of_antennas
Number of antennas (1..4) , depending on receive mode.

6.1.1.3 MultiEval

SCPI Commands :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:TOUT
CONFigure:WLAN:MEASurement<Instance>:MEValuation:CFOestimate
CONFigure:WLAN:MEASurement<Instance>:MEValuation:EMEThod
CONFigure:WLAN:MEASurement<Instance>:MEValuation:SCONdition
CONFigure:WLAN:MEASurement<Instance>:MEValuation:REPetition
CONFigure:WLAN:MEASurement<Instance>:MEValuation:MOEXception
CONFigure:WLAN:MEASurement<Instance>:MEValuation:SMODe
```

class MultiEvalCls

MultiEval commands group definition. 190 total commands, 9 Subgroups, 7 group commands

get_cfo_estimate() → CfoEstimation

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:CFOestimate
value: enums.CfoEstimation = driver.configure.wlanMeas.multiEval.get_cfo_
↪estimate()
```

No command help available

return
cfo_est: No help available

get_emethod() → EvmMethod

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:EMEThod
value: enums.EvmMethod = driver.configure.wlanMeas.multiEval.get_emethod()
```

This parameter is relevant for 802.11b signals only. It selects the EVM measurement method - according to standard IEEE Std 802.11-2007, IEEE Std 802.11b-1999, or according to standard IEEE Std 802.11-2016.

return
evm_method_11_b: No help available

get_mo_exception() → bool

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:MOEXception
value: bool = driver.configure.wlanMeas.multiEval.get_mo_exception()
```

Specifies whether measurement results that the CMP180 identifies as faulty or inaccurate are rejected.

return
meas_on_exception: OFF: Faulty results are rejected. ON: Results are never rejected.

get_repetition() → Repeat

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:REPetition
value: enums.Repeat = driver.configure.wlanMeas.multiEval.get_repetition()
```

Specifies the repetition mode of the measurement. The repetition mode specifies whether the measurement is stopped after a single shot or repeated continuously. Use CONFigure:...:MEAS<i>:...:SCOunt to determine the number of measurement intervals per single shot.

return
repetition: SINGleshot: Single-shot measurement CONTinuous: Continuous measurement

get_scondition() → StopCondition

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:SCONdition
value: enums.StopCondition = driver.configure.wlanMeas.multiEval.get_
↪scondition()
```

Qualifies whether the measurement is stopped after a failed limit check or continued. SLFail means that the measurement is stopped and reaches the RDY state when one of the results exceeds the limits.

return
stop_condition: NONE: Continue measurement irrespective of the limit check. SLFail: Stop measurement on limit failure.

get_smode() → SynchroMode

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:SMODe
value: enums.SynchroMode = driver.configure.wlanMeas.multiEval.get_smode()


    INTRO_CMD_HELP: Sets the synchronization mode:

    - Normal: synchronization according to preamble detection
    - Tolerant: synchronization with the second part of the preamble when the␣
↪first part cannot be detected

    :return: synchronization_mode: No help available
```

get_timeout() → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:TOUT
value: float = driver.configure.wlanMeas.multiEval.get_timeout()
```

Defines a timeout for the measurement. The timer is started when the measurement is initiated via a READ or INIT command. It is not started if the measurement is initiated manually. When the measurement has completed the first measurement cycle (first single shot) , the statistical depth is reached and the timer is reset. If the first measurement cycle has not been completed when the timer expires, the measurement is stopped. The measurement state changes to RDY. The reliability indicator is set to 1, indicating that a measurement timeout occurred. Still running READ, FETCh or CALCulate commands are completed, returning the available results. At least for some results, there are no values at all or the statistical depth has

not been reached. A timeout of 0 s corresponds to an infinite measurement timeout. The measurement of a DSSS signal with low data rate and large payload sizes can take up to 40 s. Set the measurement timeout to an adequate value, e.g. to 60 s.

return
tcd_timeout: No help available

set_cfo_estimate(*cfo_est: CfoEstimation*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:CFOestimate
driver.configure.wlanMeas.multiEval.set_cfo_estimate(cfo_est = enums.
↪CfoEstimation.FULLpacket)
```

No command help available

param cfo_est
No help available

set_emethod(*evm_method_11_b: EvmMethod*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:EMEThod
driver.configure.wlanMeas.multiEval.set_emethod(evm_method_11_b = enums.
↪EvmMethod.ST1999)
```

This parameter is relevant for 802.11b signals only. It selects the EVM measurement method - according to standard IEEE Std 802.11-2007, IEEE Std 802.11b-1999, or according to standard IEEE Std 802.11-2016.

param evm_method_11_b
No help available

set_mo_exception(*meas_on_exception: bool*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:MOEXception
driver.configure.wlanMeas.multiEval.set_mo_exception(meas_on_exception = False)
```

Specifies whether measurement results that the CMP180 identifies as faulty or inaccurate are rejected.

param meas_on_exception
OFF: Faulty results are rejected. ON: Results are never rejected.

set_repetition(*repetition: Repeat*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:REPetition
driver.configure.wlanMeas.multiEval.set_repetition(repetition = enums.Repeat.
↪CONTinuous)
```

Specifies the repetition mode of the measurement. The repetition mode specifies whether the measurement is stopped after a single shot or repeated continuously. Use CONFigure:..:MEAS<i>:...:SCOunt to determine the number of measurement intervals per single shot.

param repetition
SINGleshot: Single-shot measurement CONTinuous: Continuous measurement

set_scondition(*stop_condition: StopCondition*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:SCONdition
driver.configure.wlanMeas.multiEval.set_scondition(stop_condition = enums.
↪StopCondition.NONE)
```

Qualifies whether the measurement is stopped after a failed limit check or continued. SLFail means that the measurement is stopped and reaches the RDY state when one of the results exceeds the limits.

param stop_condition
NONE: Continue measurement irrespective of the limit check. SLFail: Stop measurement on limit failure.

set_smode(*synchronization_mode: SynchroMode*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:SMODe
driver.configure.wlanMeas.multiEval.set_smode(synchronization_mode = enums.
↪SynchroMode.NORMal)


    INTRO_CMD_HELP: Sets the synchronization mode:

    - Normal: synchronization according to preamble detection
    - Tolerant: synchronization with the second part of the preamble when the␣
↪first part cannot be detected

    :param synchronization_mode: No help available
```

set_timeout(*tcd_timeout: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:TOUT
driver.configure.wlanMeas.multiEval.set_timeout(tcd_timeout = 1.0)
```

Defines a timeout for the measurement. The timer is started when the measurement is initiated via a READ or INIT command. It is not started if the measurement is initiated manually. When the measurement has completed the first measurement cycle (first single shot) , the statistical depth is reached and the timer is reset. If the first measurement cycle has not been completed when the timer expires, the measurement is stopped. The measurement state changes to RDY. The reliability indicator is set to 1, indicating that a measurement timeout occurred. Still running READ, FETCh or CALCulate commands are completed, returning the available results. At least for some results, there are no values at all or the statistical depth has not been reached. A timeout of 0 s corresponds to an infinite measurement timeout. The measurement of a DSSS signal with low data rate and large payload sizes can take up to 40 s. Set the measurement timeout to an adequate value, e.g. to 60 s.

param tcd_timeout
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.clone()
```

Subgroups

6.1.1.3.1 Compensation

SCPI Commands :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:COMPensation:CESTimation
CONFigure:WLAN:MEASurement<Instance>:MEValuation:COMPensation:SMOothing
CONFigure:WLAN:MEASurement<Instance>:MEValuation:COMPensation:NCANcel
```

class CompensationCls

Compensation commands group definition. 8 total commands, 3 Subgroups, 3 group commands

get_cestimation() → ChannelEstimation

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:COMPensation:CESTimation
value: enums.ChannelEstimation = driver.configure.wlanMeas.multiEval.
↪compensation.get_cestimation()
```

Specifies whether the channel estimation is done in the payload or preamble.

return
channel_estimation: PAYLoad: Channel estimation in payload and preamble PREamble: Channel estimation in preamble only

get_ncancel() → bool

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:COMPensation:NCANcel
value: bool = driver.configure.wlanMeas.multiEval.compensation.get_ncancel()
```

Enables the noise cancelation in the CMP180 to improve EVM for channel bandwidths 160 MHz.

return
noise_cancel: No help available

get_smoothing() → bool

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:COMPensation:SMOothing
value: bool = driver.configure.wlanMeas.multiEval.compensation.get_smoothing()
```

Enables the smoothing applied to the channel estimation to improve EVM.

return
smoothing: No help available

set_cestimation(*channel_estimation: ChannelEstimation*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:COMPensation:CESTimation
driver.configure.wlanMeas.multiEval.compensation.set_cestimation(channel_
↪estimation = enums.ChannelEstimation.PAYLoad)
```

Specifies whether the channel estimation is done in the payload or preamble.

param channel_estimation
PAYLoad: Channel estimation in payload and preamble PREamble: Channel estimation in preamble only

set_ncancel(*noise_cancel: bool*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:COMPensation:NCANcel
driver.configure.wlanMeas.multiEval.compensation.set_ncancel(noise_cancel =␣
↪False)
```

Enables the noise cancelation in the CMP180 to improve EVM for channel bandwidths 160 MHz.

param noise_cancel
No help available

set_smoothing(*smoothing: bool*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:COMPensation:SMOothing
driver.configure.wlanMeas.multiEval.compensation.set_smoothing(smoothing =␣
↪False)
```

Enables the smoothing applied to the channel estimation to improve EVM.

param smoothing
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.compensation.clone()
```

Subgroups

6.1.1.3.1.1 EfTaps

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:COMPensation:EFTaps
```

class EfTapsCls

EfTaps commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class EfTapsStruct

Response structure. Fields:

- Equalizer_Filter_Taps_Enable: bool: No parameter help available

- Equalizer_Filter_Taps_Value: int: No parameter help available

get() → EfTapsStruct

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:COMPensation:EFTaps
value: EfTapsStruct = driver.configure.wlanMeas.multiEval.compensation.efTaps.
↪get()
```

This command is relevant for DSSS signals only. It determines if and how accurately the transmit filter is estimated.

return
structure: for return value, see the help for EfTapsStruct structure arguments.

set(*equalizer_filter_taps_enable: bool*, *equalizer_filter_taps_value: int = None*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:COMPensation:EFTaps
driver.configure.wlanMeas.multiEval.compensation.efTaps.set(equalizer_filter_
↪taps_enable = False, equalizer_filter_taps_value = 1)
```

This command is relevant for DSSS signals only. It determines if and how accurately the transmit filter is estimated.

param equalizer_filter_taps_enable
No help available

param equalizer_filter_taps_value
No help available

6.1.1.3.1.2 SkipSymbols

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:COMPensation:SKIPsymbols
```

class SkipSymbolsCls

SkipSymbols commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class SkipSymbolsStruct

Response structure. Fields:

- Skip_Symbols_Head: int: Number of heading symbols to be skipped.
- Skip_Symbols_Tail: int: Number of tailing symbols to be skipped.

get() → SkipSymbolsStruct

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:COMPensation:SKIPsymbols
value: SkipSymbolsStruct = driver.configure.wlanMeas.multiEval.compensation.
↪skipSymbols.get()
```

Defines how many head and tail symbols are excluded from OFDM modulation measurements.

return
structure: for return value, see the help for SkipSymbolsStruct structure arguments.

set(*skip_symbols_head: int*, *skip_symbols_tail: int*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:COMPensation:SKIPsymbols
driver.configure.wlanMeas.multiEval.compensation.skipSymbols.set(skip_symbols_
↪head = 1, skip_symbols_tail = 1)
```

Defines how many head and tail symbols are excluded from OFDM modulation measurements.

param skip_symbols_head
Number of heading symbols to be skipped.

param skip_symbols_tail
Number of tailing symbols to be skipped.

6.1.1.3.1.3 Tracking

SCPI Commands :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:COMPensation:TRACking:PHASe
CONFigure:WLAN:MEASurement<Instance>:MEValuation:COMPensation:TRACking:TIMing
CONFigure:WLAN:MEASurement<Instance>:MEValuation:COMPensation:TRACking:LEVel
```

class TrackingCls

Tracking commands group definition. 3 total commands, 0 Subgroups, 3 group commands

get_level() → bool

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:COMPensation:TRACking:LEVel
value: bool = driver.configure.wlanMeas.multiEval.compensation.tracking.get_
↪level()
```

Activate or deactivate level tracking. With enabled tracking, fluctuations are compensated.

return
level: OFF: Tracking disabled ON: Tracking enabled

get_phase() → bool

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:COMPensation:TRACking:PHASe
value: bool = driver.configure.wlanMeas.multiEval.compensation.tracking.get_
↪phase()
```

Activate or deactivate phase tracking. With enabled tracking, fluctuations are compensated.

return
phase: OFF: Tracking disabled ON: Tracking enabled

get_timing() → bool

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:COMPensation:TRACking:TIMing
value: bool = driver.configure.wlanMeas.multiEval.compensation.tracking.get_
↪timing()
```

Activate or deactivate timing tracking. With enabled tracking, fluctuations are compensated.

return
timing: OFF: Tracking disabled ON: Tracking enabled

set_level(*level: bool*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:COMPensation:TRACking:LEVel
driver.configure.wlanMeas.multiEval.compensation.tracking.set_level(level =␣
↪False)
```

Activate or deactivate level tracking. With enabled tracking, fluctuations are compensated.

param level
OFF: Tracking disabled ON: Tracking enabled

set_phase(*phase: bool*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:COMPensation:TRACking:PHASe
driver.configure.wlanMeas.multiEval.compensation.tracking.set_phase(phase =␣
↪False)
```

Activate or deactivate phase tracking. With enabled tracking, fluctuations are compensated.

param phase
OFF: Tracking disabled ON: Tracking enabled

set_timing(*timing: bool*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:COMPensation:TRACking:TIMing
driver.configure.wlanMeas.multiEval.compensation.tracking.set_timing(timing =␣
↪False)
```

Activate or deactivate timing tracking. With enabled tracking, fluctuations are compensated.

param timing
OFF: Tracking disabled ON: Tracking enabled

6.1.1.3.2 Demod

class DemodCls

Demod commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.demod.clone()
```

Subgroups

6.1.1.3.2.1 Fft

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:DEMod:FFT:OFFSet
```

class FftCls

Fft commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_offset() → FftOffset

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:DEMod:FFT:OFFSet
value: enums.FftOffset = driver.configure.wlanMeas.multiEval.demod.fft.get_
↪offset()
```

Sets the FFT start offset for OFDM signals.

return
offset: CENT: Guard interval center used as a start offset. PEAK: Peak of fine-timing metric used to determine a start offset. AUTO: Automatic selection of the optimal start offset.

set_offset(*offset: FftOffset*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:DEMod:FFT:OFFSet
driver.configure.wlanMeas.multiEval.demod.fft.set_offset(offset = enums.
↪FftOffset.AUTO)
```

Sets the FFT start offset for OFDM signals.

param offset
CENT: Guard interval center used as a start offset. PEAK: Peak of fine-timing metric used to determine a start offset. AUTO: Automatic selection of the optimal start offset.

6.1.1.3.3 Limit

SCPI Commands :

```
CONFigure:WLAN:MEASurement<instance>:MEValuation:LIMit:UTEPower
CONFigure:WLAN:MEASurement<instance>:MEValuation:LIMit:UTERror
```

class LimitCls

Limit commands group definition. 120 total commands, 4 Subgroups, 2 group commands

get_ut_error() → bool

```
# SCPI: CONFigure:WLAN:MEASurement<instance>:MEValuation:LIMit:UTERror
value: bool = driver.configure.wlanMeas.multiEval.limit.get_ut_error()
```

No command help available

return
ute_limits: No help available

get_ute_power() → LowHigh

```
# SCPI: CONFigure:WLAN:MEASurement<instance>:MEValuation:LIMit:UTEPower
value: enums.LowHigh = driver.configure.wlanMeas.multiEval.limit.get_ute_power()
```

No command help available

return
ute_power: No help available

set_ut_error(*ute_limits: bool*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<instance>:MEValuation:LIMit:UTERror
driver.configure.wlanMeas.multiEval.limit.set_ut_error(ute_limits = False)
```

No command help available

param ute_limits
No help available

set_ute_power(*ute_power: LowHigh*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<instance>:MEValuation:LIMit:UTEPower
driver.configure.wlanMeas.multiEval.limit.set_ute_power(ute_power = enums.
↪LowHigh.HIGH)
```

No command help available

param ute_power
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.clone()
```

Subgroups

6.1.1.3.3.1 Modulation

class ModulationCls

Modulation commands group definition. 42 total commands, 7 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.modulation.clone()
```

Subgroups

6.1.1.3.3.2 Dsss

SCPI Commands :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MODulation:DSSS:EVMRms
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MODulation:DSSS:EVMPeak
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MODulation:DSSS:IQOFfset
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MODulation:DSSS:CFERror
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MODulation:DSSS:CCERror
```

class DsssCls

Dsss commands group definition. 5 total commands, 0 Subgroups, 5 group commands

get_cc_error() → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:DSSS:CCERror
value: float or bool = driver.configure.wlanMeas.multiEval.limit.modulation.
↪dsss.get_cc_error()
```

Defines and activates an upper limit for the chip clock error (transmission scheme DSSS) .

return
clock_error: (float or boolean) No help available

get_cf_error() → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:DSSS:CFERror
value: float or bool = driver.configure.wlanMeas.multiEval.limit.modulation.
↪dsss.get_cf_error()
```

Defines and activates an upper limit for the center frequency error (transmission scheme DSSS) .

return
freq_error: (float or boolean) No help available

get_evm_ems() → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:DSSS:EVMRms
value: float or bool = driver.configure.wlanMeas.multiEval.limit.modulation.
↪dsss.get_evm_ems()
```

Defines and activates upper limits for the error vector magnitude (EVM) RMS values of the data carriers (transmission scheme DSSS) .

return
evm_rms: (float or boolean) No help available

get_evm_peak() → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:DSSS:EVMPeak
value: float or bool = driver.configure.wlanMeas.multiEval.limit.modulation.
↪dsss.get_evm_peak()
```

Defines and activates upper limits for the error vector magnitude (EVM) peak values of the data carriers (transmission scheme DSSS) .

return
evm_peak: (float or boolean) No help available

get_iq_offset() → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:DSSS:IQOFfset
value: float or bool = driver.configure.wlanMeas.multiEval.limit.modulation.
↪dsss.get_iq_offset()
```

Defines and activates an upper limit for the I/Q origin offset (transmission scheme DSSS) .

return
iq_offset: (float or boolean) No help available

set_cc_error(*clock_error: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:DSSS:CCERror
driver.configure.wlanMeas.multiEval.limit.modulation.dsss.set_cc_error(clock_
↪error = 1.0)
```

Defines and activates an upper limit for the chip clock error (transmission scheme DSSS) .

param clock_error
(float or boolean) No help available

set_cf_error(*freq_error: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:DSSS:CFERror
driver.configure.wlanMeas.multiEval.limit.modulation.dsss.set_cf_error(freq_
↪error = 1.0)
```

Defines and activates an upper limit for the center frequency error (transmission scheme DSSS) .

param freq_error
(float or boolean) No help available

set_evm_ems(*evm_rms: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:DSSS:EVMRms
driver.configure.wlanMeas.multiEval.limit.modulation.dsss.set_evm_ems(evm_rms =␣
↪1.0)
```

Defines and activates upper limits for the error vector magnitude (EVM) RMS values of the data carriers (transmission scheme DSSS) .

param evm_rms
(float or boolean) No help available

set_evm_peak(*evm_peak: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:DSSS:EVMPeak
driver.configure.wlanMeas.multiEval.limit.modulation.dsss.set_evm_peak(evm_peak␣
↪= 1.0)
```

Defines and activates upper limits for the error vector magnitude (EVM) peak values of the data carriers (transmission scheme DSSS) .

param evm_peak
(float or boolean) No help available

set_iq_offset(*iq_offset: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:DSSS:IQOFfset
driver.configure.wlanMeas.multiEval.limit.modulation.dsss.set_iq_offset(iq_
↪offset = 1.0)
```

Defines and activates an upper limit for the I/Q origin offset (transmission scheme DSSS) .

param iq_offset
(float or boolean) No help available

6.1.1.3.3.3 EhtOfdm

SCPI Commands :

```
CONFigure:WLAN:MEASurement<instance>:MEValuation:LIMit:MODulation:EHTofdm:EVMall
CONFigure:WLAN:MEASurement<instance>:MEValuation:LIMit:MODulation:EHTofdm:EVMPilot
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MODulation:EHTofdm:CFERror
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MODulation:EHTofdm:SCERror
```

class EhtOfdmCls

EhtOfdm commands group definition. 6 total commands, 2 Subgroups, 4 group commands

class EvmAllStruct

Structure for setting input parameters. Fields:

- Evm_Br_12: float or bool: EVM limit for BPSK, coding rate 1/2, dual carrier modulation (DCM)
- Evm_Qr_12: float or bool: EVM limit for QPSK, coding rate 1/2 DCM
- Evm_Qr_34: float or bool: EVM limit for QPSK, coding rate 3/4
- Evm_16_Qam_12: float or bool: EVM limit for 16QAM, coding rate 1/2 DCM
- Evm_16_Qam_34: float or bool: EVM limit for 16QAM, coding rate 3/4 DCM
- Evm_64_Qam_23: float or bool: EVM limit for 64QAM, coding rate 2/3
- Evm_64_Qam_34: float or bool: EVM limit for 64QAM, coding rate 3/4
- Evm_64_Qam_56: float or bool: EVM limit for 64QAM, coding rate 5/6
- Evm_256_Qam_34: float or bool: EVM limit for 256QAM, coding rate 3/4
- Evm_256_Qam_56: float or bool: EVM limit for 256QAM, coding rate 5/6
- Evm_1024_Qam_34: float or bool: EVM limit for 1024QAM, coding rate 3/4
- Evm_1024_Qam_56: float or bool: EVM limit for 1024QAM, coding rate 5/6
- Evm_4096_Qam_34: float or bool: EVM limit for 4096QAM, coding rate 3/4
- Evm_4096_Qam_56: float or bool: EVM limit for 4096QAM, coding rate 5/6
- Evm_Bdcm: float or bool: No parameter help available
- Evm_Bdcmd_Up: float or bool: No parameter help available

get_cf_error() → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:EHTofdm:CFERror
value: float or bool = driver.configure.wlanMeas.multiEval.limit.modulation.
↪ehtOfdm.get_cf_error()
```

Defines and activates an upper limit for the center frequency error in 802.11be signals.

return
center_freq_error: (float or boolean) No help available

get_evm_all() → EvmAllStruct

```
# SCPI: CONFigure:WLAN:MEASurement<instance>
↪:MEValuation:LIMit:MODulation:EHTofdm:EVMall
value: EvmAllStruct = driver.configure.wlanMeas.multiEval.limit.modulation.
↪ehtOfdm.get_evm_all()
```

Defines and activates upper limits for the error vector magnitude (EVM) of 802.11be data carriers.

return
structure: for return value, see the help for EvmAllStruct structure arguments.

get_evm_pilot() → float

```
# SCPI: CONFigure:WLAN:MEASurement<instance>
↪:MEValuation:LIMit:MODulation:EHTofdm:EVMPilot
value: float or bool = driver.configure.wlanMeas.multiEval.limit.modulation.
↪ehtOfdm.get_evm_pilot()
```

Defines and activates an upper limit for the error vector magnitude (EVM) of the pilot carriers in 802.11be signals.

return
evm_pilot: (float or boolean) No help available

get_sc_error() → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:EHTofdm:SCERror
value: float or bool = driver.configure.wlanMeas.multiEval.limit.modulation.
↪ehtOfdm.get_sc_error()
```

Defines and activates an upper limit for the symbol clock error in 802.11be signals.

return
clock_error: (float or boolean) No help available

set_cf_error(*center_freq_error: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:EHTofdm:CFERror
driver.configure.wlanMeas.multiEval.limit.modulation.ehtOfdm.set_cf_
↪error(center_freq_error = 1.0)
```

Defines and activates an upper limit for the center frequency error in 802.11be signals.

param center_freq_error
(float or boolean) No help available

set_evm_all(*value: EvmAllStruct*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<instance>
↪:MEValuation:LIMit:MODulation:EHTofdm:EVMall
structure = driver.configure.wlanMeas.multiEval.limit.modulation.ehtOfdm.
↪EvmAllStruct()
structure.Evm_Br_12: float or bool = 1.0
structure.Evm_Qr_12: float or bool = 1.0
structure.Evm_Qr_34: float or bool = 1.0
structure.Evm_16_Qam_12: float or bool = 1.0
structure.Evm_16_Qam_34: float or bool = 1.0
structure.Evm_64_Qam_23: float or bool = 1.0
structure.Evm_64_Qam_34: float or bool = 1.0
structure.Evm_64_Qam_56: float or bool = 1.0
structure.Evm_256_Qam_34: float or bool = 1.0
structure.Evm_256_Qam_56: float or bool = 1.0
structure.Evm_1024_Qam_34: float or bool = 1.0
structure.Evm_1024_Qam_56: float or bool = 1.0
structure.Evm_4096_Qam_34: float or bool = 1.0
structure.Evm_4096_Qam_56: float or bool = 1.0
structure.Evm_Bdcm: float or bool = 1.0
structure.Evm_Bdcmd_Up: float or bool = 1.0
driver.configure.wlanMeas.multiEval.limit.modulation.ehtOfdm.set_evm_all(value␣
↪= structure)
```

Defines and activates upper limits for the error vector magnitude (EVM) of 802.11be data carriers.

param value
see the help for EvmAllStruct structure arguments.

set_evm_pilot(*evm_pilot: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<instance>
↪:MEValuation:LIMit:MODulation:EHTofdm:EVMPilot
driver.configure.wlanMeas.multiEval.limit.modulation.ehtOfdm.set_evm_pilot(evm_
↪pilot = 1.0)
```

Defines and activates an upper limit for the error vector magnitude (EVM) of the pilot carriers in 802.11be signals.

param evm_pilot
(float or boolean) No help available

set_sc_error(*clock_error: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:EHTofdm:SCERror
driver.configure.wlanMeas.multiEval.limit.modulation.ehtOfdm.set_sc_error(clock_
↪error = 1.0)
```

Defines and activates an upper limit for the symbol clock error in 802.11be signals.

param clock_error
(float or boolean) No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.modulation.ehtOfdm.clone()
```

Subgroups

6.1.1.3.3.4 CfoDistribution

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MODulation:EHTofdm:CFDistrib
```

class CfoDistributionCls

CfoDistribution commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class CfoDistributionStruct

Response structure. Fields:

- Cfo_Percentage: float or bool: Upper limit for the tolerated CFO errors (CFO exceeding the specified CFO_Frequency)
- Cfo_Frequency: float: Border value defining CFO error

get() → CfoDistributionStruct

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:EHTofdm:CFDistrib
value: CfoDistributionStruct = driver.configure.wlanMeas.multiEval.limit.
↪modulation.ehtOfdm.cfoDistribution.get()
```

Configure the limit of carrier frequency offset (CFO) error distribution for EHT modulation measurements. Exceeding the limit has no impact on the stop On Limit Failure condition or out of tolerance counter.

return
structure: for return value, see the help for CfoDistributionStruct structure arguments.

set(*cfo_percentage: float*, *cfo_frequency: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:EHTofdm:CFDistrib
driver.configure.wlanMeas.multiEval.limit.modulation.ehtOfdm.cfoDistribution.
↪set(cfo_percentage = 1.0, cfo_frequency = 1.0)
```

Configure the limit of carrier frequency offset (CFO) error distribution for EHT modulation measurements. Exceeding the limit has no impact on the stop On Limit Failure condition or out of tolerance counter.

param cfo_percentage
(float or boolean) Upper limit for the tolerated CFO errors (CFO exceeding the specified CFO_Frequency)

param cfo_frequency
Border value defining CFO error

6.1.1.3.3.5 IqOffset

class IqOffsetCls

IqOffset commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.modulation.ehtOfdm.iqOffset.clone()
```

Subgroups

6.1.1.3.3.6 Bw<BandwidthG>

RepCap Settings

```
# Range: Bw5 .. Bw320
rc = driver.configure.wlanMeas.multiEval.limit.modulation.ehtOfdm.iqOffset.bw.repcap_
↪bandwidthG_get()
driver.configure.wlanMeas.multiEval.limit.modulation.ehtOfdm.iqOffset.bw.repcap_
↪bandwidthG_set(repcap.BandwidthG.Bw5)
```

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MODulation:EHTofdm:IQOFfset:BW<BW>
```

class BwCls

Bw commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: BandwidthG, default value after init: BandwidthG.Bw5

class BwStruct

Response structure. Fields:

- Offset_Value_Rel: float or bool: Relative limit
- Offset_Value_Abs: float or bool: Absolute limit

get(*bandwidthG=BandwidthG.Default*) → BwStruct

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:EHTofdm:IQOFfset:BW<BW>
value: BwStruct = driver.configure.wlanMeas.multiEval.limit.modulation.ehtOfdm.
↪iqOffset.bw.get(bandwidthG = repcap.BandwidthG.Default)
```

Defines and activates upper limits for the I/Q origin offset for 802.11be and channel bandwidth <BW>.

param bandwidthG
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

return
structure: for return value, see the help for BwStruct structure arguments.

set(*offset_value_rel: float*, *offset_value_abs: float = None*, *bandwidthG=BandwidthG.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:EHTofdm:IQOFfset:BW<BW>
driver.configure.wlanMeas.multiEval.limit.modulation.ehtOfdm.iqOffset.bw.
↪set(offset_value_rel = 1.0, offset_value_abs = 1.0, bandwidthG = repcap.
↪BandwidthG.Default)
```

Defines and activates upper limits for the I/Q origin offset for 802.11be and channel bandwidth <BW>.

param offset_value_rel
(float or boolean) Relative limit

param offset_value_abs
(float or boolean) Absolute limit

param bandwidthG
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.modulation.ehtOfdm.iqOffset.bw.clone()
```

6.1.1.3.3.7 HeOfdm

SCPI Commands :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MODulation:HEOFdm:CFERror
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MODulation:HEOFdm:SCERror
```

class HeOfdmCls

HeOfdm commands group definition. 11 total commands, 4 Subgroups, 2 group commands

get_cf_error() → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:HEOFdm:CFERror
value: float or bool = driver.configure.wlanMeas.multiEval.limit.modulation.
↪heOfdm.get_cf_error()
```

Defines and activates an upper limit for the center frequency error in 802.11ax signals.

return
center_freq_error: (float or boolean) No help available

get_sc_error() → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:HEOFdm:SCERror
value: float or bool = driver.configure.wlanMeas.multiEval.limit.modulation.
↪heOfdm.get_sc_error()
```

Defines and activates an upper limit for the symbol clock error in 802.11ax signals.

return
clock_error: (float or boolean) No help available

set_cf_error(*center_freq_error: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:HEOFdm:CFERror
driver.configure.wlanMeas.multiEval.limit.modulation.heOfdm.set_cf_error(center_
↪freq_error = 1.0)
```

Defines and activates an upper limit for the center frequency error in 802.11ax signals.

param center_freq_error
(float or boolean) No help available

set_sc_error(*clock_error: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:HEOFdm:SCERror
driver.configure.wlanMeas.multiEval.limit.modulation.heOfdm.set_sc_error(clock_
↪error = 1.0)
```

Defines and activates an upper limit for the symbol clock error in 802.11ax signals.

param clock_error
(float or boolean) No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.modulation.heOfdm.clone()
```

Subgroups

6.1.1.3.3.8 CfoDistribution

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MODulation:HEOFdm:CFDistrib
```

class CfoDistributionCls

CfoDistribution commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class CfoDistributionStruct

Response structure. Fields:

- Cfo_Percentage: float or bool: Upper limit for the tolerated CFO errors (CFO exceeding the specified CFO_Frequency)
- Cfo_Frequency: float: Border value defining CFO error

get() → CfoDistributionStruct

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:HEOFdm:CFDistrib
value: CfoDistributionStruct = driver.configure.wlanMeas.multiEval.limit.
↪modulation.heOfdm.cfoDistribution.get()
```

Configure the limit of carrier frequency offset (CFO) error distribution for HE modulation measurements. Exceeding the limit has no impact on the stop On Limit Failure condition or out of tolerance counter.

return
structure: for return value, see the help for CfoDistributionStruct structure arguments.

set(*cfo_percentage: float*, *cfo_frequency: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:HEOFdm:CFDistrib
driver.configure.wlanMeas.multiEval.limit.modulation.heOfdm.cfoDistribution.
↪set(cfo_percentage = 1.0, cfo_frequency = 1.0)
```

Configure the limit of carrier frequency offset (CFO) error distribution for HE modulation measurements. Exceeding the limit has no impact on the stop On Limit Failure condition or out of tolerance counter.

param cfo_percentage
(float or boolean) Upper limit for the tolerated CFO errors (CFO exceeding the specified CFO_Frequency)

param cfo_frequency
Border value defining CFO error

6.1.1.3.3.9 EvmAll

SCPI Commands :

```
CONFigure:WLAN:MEASurement<instance>
↪:MEValuation:LIMit:MODulation:HEOFdm:EVMall:TBCoderate
CONFigure:WLAN:MEASurement<instance>:MEValuation:LIMit:MODulation:HEOFdm:EVMall:TBHigh
CONFigure:WLAN:MEASurement<instance>:MEValuation:LIMit:MODulation:HEOFdm:EVMall:TBLow
CONFigure:WLAN:MEASurement<instance>:MEValuation:LIMit:MODulation:HEOFdm:EVMall
```

class EvmAllCls

EvmAll commands group definition. 4 total commands, 0 Subgroups, 4 group commands

class TbCoderateStruct

Structure for setting input parameters. Fields:

- Cr_Bpsk: enums.Coderate: Coding rate for BPSK modulation format CR14dcm: 1/4 DCM (coding rate 1/2 with DCM) CR38dcm: 3/8 DCM (coding rate 3/4 with DCM) CR12: 1/2 (coding rate 1/2 without DCM) CR23: 2/3 (coding rate 2/3 without DCM) CR34: 3/4 (coding rate 3/4 without DCM) CR56: 5/6 (coding rate 5/6 without DCM)
- Cr_Qpsk: enums.Coderate: No parameter help available
- Cr_16_Qam: enums.Coderate: No parameter help available
- Cr_64_Qam: enums.Coderate: No parameter help available
- Cr_256_Qam: enums.Coderate: No parameter help available
- Cr_1024_Qam: enums.Coderate: No parameter help available

class TbHighStruct

Structure for setting input parameters. Fields:

- Evm_Bpsk: float or bool: EVM limit for BPSK
- Evm_Qpsk: float or bool: EVM limit for QPSK
- Evm_16_Qam: float or bool: EVM limit for 16QAM
- Evm_64_Qam: float or bool: EVM limit for 64QAM
- Evm_256_Qam: float or bool: EVM limit for 256QAM
- Evm_1024_Qam: float or bool: EVM limit for 1024QAM

class TbLowStruct

Structure for setting input parameters. Fields:

- Evm_Bpsk: float or bool: EVM limit for BPSK

- Evm_Qpsk: float or bool: EVM limit for QPSK
- Evm_16_Qam: float or bool: EVM limit for 16QAM
- Evm_64_Qam: float or bool: EVM limit for 64QAM
- Evm_256_Qam: float or bool: EVM limit for 256QAM
- Evm_1024_Qam: float or bool: EVM limit for 1024QAM

class ValueStruct

Structure for setting input parameters. Contains optional set arguments. Fields:

- Evm_Br_14: float or bool: Limits for BPSK, coding rate 1/4, dual carrier modulation (DCM)
- Evm_Br_12: float or bool: Limits for BPSK, coding rate 1/2
- Evm_Qr_14: float or bool: Limits for QPSK, coding rate 1/4 DCM
- Evm_Qr_12: float or bool: Limits for QPSK, coding rate 1/2
- Evm_Qr_34: float or bool: Limits for QPSK, coding rate 3/4
- Evm_16_Qam_14: float or bool: Limits for 16QAM, coding rate 1/4 DCM
- Evm_16_Qam_38: float or bool: Limits for 16QAM, coding rate 3/8 DCM
- Evm_16_Qam_12: float or bool: Limits for 16QAM, coding rate 1/2
- Evm_16_Qam_34: float or bool: Limits for 16QAM, coding rate 3/4
- Evm_64_Qam_23: float or bool: Limits for 64QAM, coding rate 2/3
- Evm_64_Qam_34: float or bool: Limits for 64QAM, coding rate 3/4
- Evm_64_Qam_56: float or bool: Limits for 64QAM, coding rate 5/6
- Evm_256_Qam_34: float or bool: Limits for 256QAM, coding rate 3/4
- Evm_256_Qam_56: float or bool: Limits for 256QAM, coding rate 5/6
- Evm_1024_Qam_34: float or bool: Optional setting parameter. Limits for 1024QAM, coding rate 3/4
- Evm_1024_Qam_56: float or bool: Optional setting parameter. Limits for 1024QAM, coding rate 5/6

get_tb_coderate() → TbCoderateStruct

```
# SCPI: CONFigure:WLAN:MEASurement<instance>
↪:MEValuation:LIMit:MODulation:HEOFdm:EVMall:TBCoderate
value: TbCoderateStruct = driver.configure.wlanMeas.multiEval.limit.modulation.
↪heOfdm.evmAll.get_tb_coderate()
```

Specifies the coding rate of HE TB PPDU per modulation format, used for the calculation of unused tone error limit line.

return
structure: for return value, see the help for TbCoderateStruct structure arguments.

get_tb_high() → TbHighStruct

```
# SCPI: CONFigure:WLAN:MEASurement<instance>
↪:MEValuation:LIMit:MODulation:HEOFdm:EVMall:TBHigh
value: TbHighStruct = driver.configure.wlanMeas.multiEval.limit.modulation.
↪heOfdm.evmAll.get_tb_high()
```

Sets EVM limits for HE TB PPDU when transmit power is larger than the maximum power of MCS 7. The default values are in line with standard IEEE Std 802.11ax-2021, table 27-49 Allowed relative constellation error versus constellation size and coding rate.

return
structure: for return value, see the help for TbHighStruct structure arguments.

get_tb_low() → TbLowStruct

```
# SCPI: CONFigure:WLAN:MEASurement<instance>
↪:MEValuation:LIMit:MODulation:HEOFdm:EVMall:TBLow
value: TbLowStruct = driver.configure.wlanMeas.multiEval.limit.modulation.
↪heOfdm.evmAll.get_tb_low()
```

Sets EVM limits for HE TB PPDU when transmit power is less than or equal to the maximum power of MCS 7. The default values are in line with standard IEEE Std 802.11ax-2021, table 27-49 Allowed relative constellation error versus constellation size and coding rate.

return
structure: for return value, see the help for TbLowStruct structure arguments.

get_value() → ValueStruct

```
# SCPI: CONFigure:WLAN:MEASurement<instance>
↪:MEValuation:LIMit:MODulation:HEOFdm:EVMall
value: ValueStruct = driver.configure.wlanMeas.multiEval.limit.modulation.
↪heOfdm.evmAll.get_value()
```

Defines and activates upper limits for the error vector magnitude (EVM) of 802.11ax data carriers.

return
structure: for return value, see the help for ValueStruct structure arguments.

set_tb_coderate(*value: TbCoderateStruct*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<instance>
↪:MEValuation:LIMit:MODulation:HEOFdm:EVMall:TBCoderate
structure = driver.configure.wlanMeas.multiEval.limit.modulation.heOfdm.evmAll.
↪TbCoderateStruct()
structure.Cr_Bpsk: enums.Coderate = enums.Coderate.AUTO
structure.Cr_Qpsk: enums.Coderate = enums.Coderate.AUTO
structure.Cr_16_Qam: enums.Coderate = enums.Coderate.AUTO
structure.Cr_64_Qam: enums.Coderate = enums.Coderate.AUTO
structure.Cr_256_Qam: enums.Coderate = enums.Coderate.AUTO
structure.Cr_1024_Qam: enums.Coderate = enums.Coderate.AUTO
driver.configure.wlanMeas.multiEval.limit.modulation.heOfdm.evmAll.set_tb_
↪coderate(value = structure)
```

Specifies the coding rate of HE TB PPDU per modulation format, used for the calculation of unused tone error limit line.

param value
see the help for TbCoderateStruct structure arguments.

set_tb_high(*value: TbHighStruct*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<instance>
↪:MEValuation:LIMit:MODulation:HEOFdm:EVMall:TBHigh
structure = driver.configure.wlanMeas.multiEval.limit.modulation.heOfdm.evmAll.
↪TbHighStruct()
structure.Evm_Bpsk: float or bool = 1.0
structure.Evm_Qpsk: float or bool = 1.0
structure.Evm_16_Qam: float or bool = 1.0
structure.Evm_64_Qam: float or bool = 1.0
structure.Evm_256_Qam: float or bool = 1.0
structure.Evm_1024_Qam: float or bool = 1.0
driver.configure.wlanMeas.multiEval.limit.modulation.heOfdm.evmAll.set_tb_
↪high(value = structure)
```

Sets EVM limits for HE TB PPDU when transmit power is larger than the maximum power of MCS 7. The default values are in line with standard IEEE Std 802.11ax-2021, table 27-49 Allowed relative constellation error versus constellation size and coding rate.

param value
see the help for TbHighStruct structure arguments.

set_tb_low(*value: TbLowStruct*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<instance>
↪:MEValuation:LIMit:MODulation:HEOFdm:EVMall:TBLow
structure = driver.configure.wlanMeas.multiEval.limit.modulation.heOfdm.evmAll.
↪TbLowStruct()
structure.Evm_Bpsk: float or bool = 1.0
structure.Evm_Qpsk: float or bool = 1.0
structure.Evm_16_Qam: float or bool = 1.0
structure.Evm_64_Qam: float or bool = 1.0
structure.Evm_256_Qam: float or bool = 1.0
structure.Evm_1024_Qam: float or bool = 1.0
driver.configure.wlanMeas.multiEval.limit.modulation.heOfdm.evmAll.set_tb_
↪low(value = structure)
```

Sets EVM limits for HE TB PPDU when transmit power is less than or equal to the maximum power of MCS 7. The default values are in line with standard IEEE Std 802.11ax-2021, table 27-49 Allowed relative constellation error versus constellation size and coding rate.

param value
see the help for TbLowStruct structure arguments.

set_value(*value: ValueStruct*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<instance>
↪:MEValuation:LIMit:MODulation:HEOFdm:EVMall
structure = driver.configure.wlanMeas.multiEval.limit.modulation.heOfdm.evmAll.
↪ValueStruct()
structure.Evm_Br_14: float or bool = 1.0
structure.Evm_Br_12: float or bool = 1.0
structure.Evm_Qr_14: float or bool = 1.0
structure.Evm_Qr_12: float or bool = 1.0
structure.Evm_Qr_34: float or bool = 1.0
structure.Evm_16_Qam_14: float or bool = 1.0
structure.Evm_16_Qam_38: float or bool = 1.0
```

(continues on next page)

(continued from previous page)

```
structure.Evm_16_Qam_12: float or bool = 1.0
structure.Evm_16_Qam_34: float or bool = 1.0
structure.Evm_64_Qam_23: float or bool = 1.0
structure.Evm_64_Qam_34: float or bool = 1.0
structure.Evm_64_Qam_56: float or bool = 1.0
structure.Evm_256_Qam_34: float or bool = 1.0
structure.Evm_256_Qam_56: float or bool = 1.0
structure.Evm_1024_Qam_34: float or bool = 1.0
structure.Evm_1024_Qam_56: float or bool = 1.0
driver.configure.wlanMeas.multiEval.limit.modulation.heOfdm.evmAll.set_
↪value(value = structure)
```

Defines and activates upper limits for the error vector magnitude (EVM) of 802.11ax data carriers.

param value
see the help for ValueStruct structure arguments.

6.1.1.3.3.10 EvmPilot

SCPI Commands :

```
CONFigure:WLAN:MEASurement<instance>:MEValuation:LIMit:MODulation:HEOFdm:EVMPilot:TBHigh
CONFigure:WLAN:MEASurement<instance>:MEValuation:LIMit:MODulation:HEOFdm:EVMPilot:TBLow
CONFigure:WLAN:MEASurement<instance>:MEValuation:LIMit:MODulation:HEOFdm:EVMPilot
```

class EvmPilotCls

EvmPilot commands group definition. 3 total commands, 0 Subgroups, 3 group commands

get_tb_high() → float

```
# SCPI: CONFigure:WLAN:MEASurement<instance>
↪:MEValuation:LIMit:MODulation:HEOFdm:EVMPilot:TBHigh
value: float or bool = driver.configure.wlanMeas.multiEval.limit.modulation.
↪heOfdm.evmPilot.get_tb_high()
```

Sets EVM limits for a pilot subcarrier in 802.11ax trigger-based signals, when transmit power is larger than the maximum power of MCS 7.

return
evm_pilot: (float or boolean) No help available

get_tb_low() → float

```
# SCPI: CONFigure:WLAN:MEASurement<instance>
↪:MEValuation:LIMit:MODulation:HEOFdm:EVMPilot:TBLow
value: float or bool = driver.configure.wlanMeas.multiEval.limit.modulation.
↪heOfdm.evmPilot.get_tb_low()
```

Sets EVM limits for a pilot subcarrier in 802.11ax trigger-based signals, when transmit power is less than or equal to the maximum power of MCS 7.

return
evm_pilot: (float or boolean) No help available

get_value() → float

```
# SCPI: CONFigure:WLAN:MEASurement<instance>
↪:MEValuation:LIMit:MODulation:HEOFdm:EVMPilot
value: float or bool = driver.configure.wlanMeas.multiEval.limit.modulation.
↪heOfdm.evmPilot.get_value()
```

Defines and activates an upper limit for the error vector magnitude (EVM) of the pilot carriers in 802.11ax signals.

return
evm_pilot: (float or boolean) No help available

set_tb_high(*evm_pilot: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<instance>
↪:MEValuation:LIMit:MODulation:HEOFdm:EVMPilot:TBHigh
driver.configure.wlanMeas.multiEval.limit.modulation.heOfdm.evmPilot.set_tb_
↪high(evm_pilot = 1.0)
```

Sets EVM limits for a pilot subcarrier in 802.11ax trigger-based signals, when transmit power is larger than the maximum power of MCS 7.

param evm_pilot
(float or boolean) No help available

set_tb_low(*evm_pilot: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<instance>
↪:MEValuation:LIMit:MODulation:HEOFdm:EVMPilot:TBLow
driver.configure.wlanMeas.multiEval.limit.modulation.heOfdm.evmPilot.set_tb_
↪low(evm_pilot = 1.0)
```

Sets EVM limits for a pilot subcarrier in 802.11ax trigger-based signals, when transmit power is less than or equal to the maximum power of MCS 7.

param evm_pilot
(float or boolean) No help available

set_value(*evm_pilot: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<instance>
↪:MEValuation:LIMit:MODulation:HEOFdm:EVMPilot
driver.configure.wlanMeas.multiEval.limit.modulation.heOfdm.evmPilot.set_
↪value(evm_pilot = 1.0)
```

Defines and activates an upper limit for the error vector magnitude (EVM) of the pilot carriers in 802.11ax signals.

param evm_pilot
(float or boolean) No help available

6.1.1.3.3.11 IqOffset

class IqOffsetCls

IqOffset commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.modulation.heOfdm.iqOffset.clone()
```

Subgroups

6.1.1.3.3.12 Bw<BandwidthE>

RepCap Settings

```
# Range: Bw5 .. Bw8080
rc = driver.configure.wlanMeas.multiEval.limit.modulation.heOfdm.iqOffset.bw.repcap_
↪bandwidthE_get()
driver.configure.wlanMeas.multiEval.limit.modulation.heOfdm.iqOffset.bw.repcap_
↪bandwidthE_set(repcap.BandwidthE.Bw5)
```

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MODulation:HEOFdm:IQOFfset:BW<BW>
```

class BwCls

Bw commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: BandwidthE, default value after init: BandwidthE.Bw5

class BwStruct

Response structure. Fields:

- Offset_Value_Rel: float or bool: Relative limit
- Offset_Value_Abs: float or bool: Absolute limit

get(*bandwidthE=BandwidthE.Default*) → BwStruct

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:HEOFdm:IQOFfset:BW<BW>
value: BwStruct = driver.configure.wlanMeas.multiEval.limit.modulation.heOfdm.
↪iqOffset.bw.get(bandwidthE = repcap.BandwidthE.Default)
```

Defines and activates upper limits for the I/Q origin offset for 802.11ax and channel bandwidth <BW>.

param bandwidthE
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

return
structure: for return value, see the help for BwStruct structure arguments.

set(*offset_value_rel: float*, *offset_value_abs: float = None*, *bandwidthE=BandwidthE.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:HEOFdm:IQOFfset:BW<BW>
driver.configure.wlanMeas.multiEval.limit.modulation.heOfdm.iqOffset.bw.
↪set(offset_value_rel = 1.0, offset_value_abs = 1.0, bandwidthE = repcap.
↪BandwidthE.Default)
```

Defines and activates upper limits for the I/Q origin offset for 802.11ax and channel bandwidth <BW>.

param offset_value_rel
(float or boolean) Relative limit

param offset_value_abs
(float or boolean) Absolute limit

param bandwidthE
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.modulation.heOfdm.iqOffset.bw.clone()
```

6.1.1.3.3.13 HtOfdm

SCPI Commands :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MODulation:HTOFdm:EVM
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MODulation:HTOFdm:EVMPilot
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MODulation:HTOFdm:CFERror
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MODulation:HTOFdm:SCERror
```

class HtOfdmCls

HtOfdm commands group definition. 5 total commands, 1 Subgroups, 4 group commands

class EvmStruct

Structure for setting input parameters. Fields:

- Evm_Br_12: float or bool: Limits for BPSK, coding rate 1/2
- Evm_Qr_12: float or bool: Limits for QPSK, coding rate 1/2
- Evm_Qr_34: float or bool: Limits for QPSK, coding rate 3/4
- Evm_Q_1_M_12: float or bool: Limits for 16QAM, coding rate 1/2
- Evm_Q_1_M_34: float or bool: Limits for 16QAM, coding rate 3/4
- Evm_Q_6_M_12: float or bool: Limits for 64QAM, coding rate 1/2
- Evm_Q_6_M_34: float or bool: Limits for 64QAM, coding rate 3/4
- Evm_Q_6_M_56: float or bool: Limits for 64QAM, coding rate 5/6

get_cf_error() → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:HTOFdm:CFERror
value: float or bool = driver.configure.wlanMeas.multiEval.limit.modulation.
↪htOfdm.get_cf_error()
```

Defines and activates an upper limit for the center frequency error (802.11n) .

return
center_freq_error: (float or boolean) No help available

get_evm() → EvmStruct

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:HTOFdm:EVM
value: EvmStruct = driver.configure.wlanMeas.multiEval.limit.modulation.htOfdm.
↪get_evm()
```

Defines and activates upper limits for the error vector magnitude (EVM) of the data carriers (802.11n) .

return
structure: for return value, see the help for EvmStruct structure arguments.

get_evm_pilot() → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:HTOFdm:EVMPilot
value: float or bool = driver.configure.wlanMeas.multiEval.limit.modulation.
↪htOfdm.get_evm_pilot()
```

Defines and activates an upper limit for the error vector magnitude (EVM) of the pilot carriers (802.11n) .

return
evm_pilot: (float or boolean) No help available

get_sc_error() → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:HTOFdm:SCERror
value: float or bool = driver.configure.wlanMeas.multiEval.limit.modulation.
↪htOfdm.get_sc_error()
```

Defines and activates an upper limit for the symbol clock error (802.11n) .

return
clock_error: (float or boolean) No help available

set_cf_error(*center_freq_error: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:HTOFdm:CFERror
driver.configure.wlanMeas.multiEval.limit.modulation.htOfdm.set_cf_error(center_
↪freq_error = 1.0)
```

Defines and activates an upper limit for the center frequency error (802.11n) .

param center_freq_error
(float or boolean) No help available

set_evm(*value: EvmStruct*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:HTOFdm:EVM
structure = driver.configure.wlanMeas.multiEval.limit.modulation.htOfdm.
↪EvmStruct()
structure.Evm_Br_12: float or bool = 1.0
structure.Evm_Qr_12: float or bool = 1.0
structure.Evm_Qr_34: float or bool = 1.0
structure.Evm_Q_1_M_12: float or bool = 1.0
structure.Evm_Q_1_M_34: float or bool = 1.0
structure.Evm_Q_6_M_12: float or bool = 1.0
structure.Evm_Q_6_M_34: float or bool = 1.0
structure.Evm_Q_6_M_56: float or bool = 1.0
driver.configure.wlanMeas.multiEval.limit.modulation.htOfdm.set_evm(value =␣
↪structure)
```

Defines and activates upper limits for the error vector magnitude (EVM) of the data carriers (802.11n) .

param value
see the help for EvmStruct structure arguments.

set_evm_pilot(*evm_pilot: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:HTOFdm:EVMPilot
driver.configure.wlanMeas.multiEval.limit.modulation.htOfdm.set_evm_pilot(evm_
↪pilot = 1.0)
```

Defines and activates an upper limit for the error vector magnitude (EVM) of the pilot carriers (802.11n) .

param evm_pilot
(float or boolean) No help available

set_sc_error(*clock_error: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:HTOFdm:SCERror
driver.configure.wlanMeas.multiEval.limit.modulation.htOfdm.set_sc_error(clock_
↪error = 1.0)
```

Defines and activates an upper limit for the symbol clock error (802.11n) .

param clock_error
(float or boolean) No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.modulation.htOfdm.clone()
```

Subgroups

6.1.1.3.3.14 IqOffset

class IqOffsetCls

IqOffset commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.modulation.htOfdm.iqOffset.clone()
```

Subgroups

6.1.1.3.3.15 Bw<BandwidthC>

RepCap Settings

```
# Range: Bw5 .. Bw40
rc = driver.configure.wlanMeas.multiEval.limit.modulation.htOfdm.iqOffset.bw.repcap_
↪bandwidthC_get()
driver.configure.wlanMeas.multiEval.limit.modulation.htOfdm.iqOffset.bw.repcap_
↪bandwidthC_set(repcap.BandwidthC.Bw5)
```

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MODulation:HTOFdm:IQOFfset:BW<BW>
```

class BwCls

Bw commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: BandwidthC, default value after init: BandwidthC.Bw5

get(*bandwidthC=BandwidthC.Default*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:HTOFdm:IQOFfset:BW<BW>
value: float or bool = driver.configure.wlanMeas.multiEval.limit.modulation.
↪htOfdm.iqOffset.bw.get(bandwidthC = repcap.BandwidthC.Default)
```

Defines and activates an upper limit for the I/Q origin offset, for 802.11n and channel bandwidth <BW>.

param bandwidthC
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

return
offset_value: (float or boolean) No help available

set(*offset_value: float*, *bandwidthC=BandwidthC.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:HTOFdm:IQOFfset:BW<BW>
driver.configure.wlanMeas.multiEval.limit.modulation.htOfdm.iqOffset.bw.
↪set(offset_value = 1.0, bandwidthC = repcap.BandwidthC.Default)
```

Defines and activates an upper limit for the I/Q origin offset, for 802.11n and channel bandwidth <BW>.

param offset_value
(float or boolean) No help available

param bandwidthC
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.modulation.htOfdm.iqOffset.bw.clone()
```

6.1.1.3.3.16 Lofdm

SCPI Commands :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MODulation:LOFDm:EVM
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MODulation:LOFDm:EVMPilot
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MODulation:LOFDm:IQOFfset
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MODulation:LOFDm:CFERror
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MODulation:LOFDm:SCERror
```

class LofdmCls

Lofdm commands group definition. 5 total commands, 0 Subgroups, 5 group commands

class EvmStruct

Structure for setting input parameters. Fields:

- Evm_6_M: float or bool: Limit for data rate 6 Mbit/s
- Evm_9_M: float or bool: Limit for data rate 9 Mbit/s
- Evm_12_M: float or bool: Limit for data rate 12 Mbit/s
- Evm_18_M: float or bool: Limit for data rate 18 Mbit/s
- Evm_24_M: float or bool: Limit for data rate 24 Mbit/s
- Evm_36_M: float or bool: Limit for data rate 36 Mbit/s
- Evm_48_M: float or bool: Limit for data rate 48 Mbit/s
- Evm_54_M: float or bool: Limit for data rate 54 Mbit/s

get_cf_error() → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:LOFDm:CFERror
value: float or bool = driver.configure.wlanMeas.multiEval.limit.modulation.
↪lofdm.get_cf_error()
```

Defines and activates an upper limit for the center frequency error (802.11a/g, OFDM) .

return
center_freq_error: (float or boolean) No help available

get_evm() → EvmStruct

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:LOFDm:EVM
value: EvmStruct = driver.configure.wlanMeas.multiEval.limit.modulation.lofdm.
↪get_evm()
```

Defines and activates upper limits for the error vector magnitude (EVM) of the data carriers (802.11a/g, OFDM) .

return
structure: for return value, see the help for EvmStruct structure arguments.

get_evm_pilot() → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:LOFDm:EVMPilot
value: float or bool = driver.configure.wlanMeas.multiEval.limit.modulation.
↪lofdm.get_evm_pilot()
```

Defines and activates an upper limit for the error vector magnitude (EVM) of the pilot carriers (802.11a/g, OFDM) .

return
evm_pilot: (float or boolean) No help available

get_iq_offset() → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:LOFDm:IQOFfset
value: float or bool = driver.configure.wlanMeas.multiEval.limit.modulation.
↪lofdm.get_iq_offset()
```

Defines and activates an upper limit for the I/Q origin offset (802.11a/g, OFDM) .

return
iq_offset: (float or boolean) No help available

get_sc_error() → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:LOFDm:SCERror
value: float or bool = driver.configure.wlanMeas.multiEval.limit.modulation.
↪lofdm.get_sc_error()
```

Defines and activates an upper limit for the symbol clock error (802.11a/g, OFDM) .

return
clock_error: (float or boolean) No help available

set_cf_error(*center_freq_error: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:LOFDm:CFERror
driver.configure.wlanMeas.multiEval.limit.modulation.lofdm.set_cf_error(center_
↪freq_error = 1.0)
```

Defines and activates an upper limit for the center frequency error (802.11a/g, OFDM) .

param center_freq_error
(float or boolean) No help available

set_evm(*value: EvmStruct*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:LOFDm:EVM
structure = driver.configure.wlanMeas.multiEval.limit.modulation.lofdm.
↪EvmStruct()
structure.Evm_6_M: float or bool = 1.0
structure.Evm_9_M: float or bool = 1.0
structure.Evm_12_M: float or bool = 1.0
structure.Evm_18_M: float or bool = 1.0
structure.Evm_24_M: float or bool = 1.0
structure.Evm_36_M: float or bool = 1.0
structure.Evm_48_M: float or bool = 1.0
structure.Evm_54_M: float or bool = 1.0
driver.configure.wlanMeas.multiEval.limit.modulation.lofdm.set_evm(value =␣
↪structure)
```

Defines and activates upper limits for the error vector magnitude (EVM) of the data carriers (802.11a/g, OFDM) .

param value
see the help for EvmStruct structure arguments.

set_evm_pilot(*evm_pilot: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:LOFDm:EVMPilot
driver.configure.wlanMeas.multiEval.limit.modulation.lofdm.set_evm_pilot(evm_
↪pilot = 1.0)
```

Defines and activates an upper limit for the error vector magnitude (EVM) of the pilot carriers (802.11a/g, OFDM) .

param evm_pilot
(float or boolean) No help available

set_iq_offset(*iq_offset: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:LOFDm:IQOFfset
driver.configure.wlanMeas.multiEval.limit.modulation.lofdm.set_iq_offset(iq_
↪offset = 1.0)
```

Defines and activates an upper limit for the I/Q origin offset (802.11a/g, OFDM) .

param iq_offset
(float or boolean) No help available

set_sc_error(*clock_error: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:LOFDm:SCERror
driver.configure.wlanMeas.multiEval.limit.modulation.lofdm.set_sc_error(clock_
↪error = 1.0)
```

Defines and activates an upper limit for the symbol clock error (802.11a/g, OFDM) .

param clock_error
(float or boolean) No help available

6.1.1.3.3.17 Pofdm

SCPI Commands :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MODulation:POFDm:EVM
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MODulation:POFDm:EVMPilot
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MODulation:POFDm:IQOFfset
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MODulation:POFDm:CFERror
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MODulation:POFDm:SCERror
```

class PofdmCls

Pofdm commands group definition. 5 total commands, 0 Subgroups, 5 group commands

class EvmStruct

Structure for setting input parameters. Fields:

- Bpsk_12: float or bool: Limit for data rate BPSK modulation and coding rate 1/2
- Bpsk_34: float or bool: Limit for data rate BPSK modulation and coding rate 3/4
- Qpsk_12: float or bool: Limit for data rate QPSK modulation and coding rate 1/2
- Qpsk_34: float or bool: Limit for data rate QPSK modulation and coding rate 3/4
- Q_16_Am_12: float or bool: Limit for data rate 16QAM modulation and coding rate 1/2
- Q_16_Am_34: float or bool: Limit for data rate 16QAM modulation and coding rate 3/4
- Q_64_Am_23: float or bool: Limit for data rate 64QAM modulation and coding rate 2/3
- Q_64_Am_34: float or bool: Limit for data rate 64QAM modulation and coding rate 3/4

get_cf_error() → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:POFDm:CFERror
value: float or bool = driver.configure.wlanMeas.multiEval.limit.modulation.
↪pofdm.get_cf_error()
```

Defines and activates an upper limit for the center frequency error (802.11p) .

return
center_freq_error: (float or boolean) No help available

get_evm() → EvmStruct

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:POFDm:EVM
value: EvmStruct = driver.configure.wlanMeas.multiEval.limit.modulation.pofdm.
↪get_evm()
```

Defines and activates upper limits for the error vector magnitude (EVM) of the data carriers in 802.11p signals.

return
structure: for return value, see the help for EvmStruct structure arguments.

get_evm_pilot() → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:POFDm:EVMPilot
value: float or bool = driver.configure.wlanMeas.multiEval.limit.modulation.
↪pofdm.get_evm_pilot()
```

Defines and activates an upper limit for the error vector magnitude (EVM) of 802.11p pilot carriers.

return
evm_pilot: (float or boolean) No help available

get_iq_offset() → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:POFDm:IQOFfset
value: float or bool = driver.configure.wlanMeas.multiEval.limit.modulation.
↪pofdm.get_iq_offset()
```

Defines and activates an upper limit for the I/Q origin offset of 802.11p signals.

return
iq_offset: (float or boolean) No help available

get_sc_error() → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:POFDm:SCERror
value: float or bool = driver.configure.wlanMeas.multiEval.limit.modulation.
↪pofdm.get_sc_error()
```

Defines and activates an upper limit for the symbol clock error (802.11p) .

return
clock_error: (float or boolean) No help available

set_cf_error(*center_freq_error: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:POFDm:CFERror
driver.configure.wlanMeas.multiEval.limit.modulation.pofdm.set_cf_error(center_
↪freq_error = 1.0)
```

Defines and activates an upper limit for the center frequency error (802.11p) .

param center_freq_error
(float or boolean) No help available

set_evm(*value: EvmStruct*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:POFDm:EVM
structure = driver.configure.wlanMeas.multiEval.limit.modulation.pofdm.
↪EvmStruct()
structure.Bpsk_12: float or bool = 1.0
structure.Bpsk_34: float or bool = 1.0
structure.Qpsk_12: float or bool = 1.0
structure.Qpsk_34: float or bool = 1.0
structure.Q_16_Am_12: float or bool = 1.0
structure.Q_16_Am_34: float or bool = 1.0
structure.Q_64_Am_23: float or bool = 1.0
structure.Q_64_Am_34: float or bool = 1.0
driver.configure.wlanMeas.multiEval.limit.modulation.pofdm.set_evm(value =␣
↪structure)
```

Defines and activates upper limits for the error vector magnitude (EVM) of the data carriers in 802.11p signals.

param value
see the help for EvmStruct structure arguments.

set_evm_pilot(*evm_pilot: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:POFDm:EVMPilot
driver.configure.wlanMeas.multiEval.limit.modulation.pofdm.set_evm_pilot(evm_
↪pilot = 1.0)
```

Defines and activates an upper limit for the error vector magnitude (EVM) of 802.11p pilot carriers.

param evm_pilot
(float or boolean) No help available

set_iq_offset(*iq_offset: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:POFDm:IQOFfset
driver.configure.wlanMeas.multiEval.limit.modulation.pofdm.set_iq_offset(iq_
↪offset = 1.0)
```

Defines and activates an upper limit for the I/Q origin offset of 802.11p signals.

param iq_offset
(float or boolean) No help available

set_sc_error(*clock_error: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:POFDm:SCERror
driver.configure.wlanMeas.multiEval.limit.modulation.pofdm.set_sc_error(clock_
↪error = 1.0)
```

Defines and activates an upper limit for the symbol clock error (802.11p) .

param clock_error
(float or boolean) No help available

6.1.1.3.3.18 VhtOfdm

SCPI Commands :

```
CONFigure:WLAN:MEASurement<instance>:MEValuation:LIMit:MODulation:VHTofdm:EVMall
CONFigure:WLAN:MEASurement<instance>:MEValuation:LIMit:MODulation:VHTofdm:EVMPilot
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MODulation:VHTofdm:CFERror
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MODulation:VHTofdm:SCERror
```

class VhtOfdmCls

VhtOfdm commands group definition. 5 total commands, 1 Subgroups, 4 group commands

class EvmAllStruct

Structure for setting input parameters. Contains optional set arguments. Fields:

- Evm_Br_12: float or bool: Limits for BPSK, coding rate 1/2
- Evm_Qr_12: float or bool: Limits for QPSK, coding rate 1/2
- Evm_Qr_34: float or bool: Limits for QPSK, coding rate 3/4
- Evm_16_Qam_12: float or bool: Limits for 16QAM, coding rate 1/2
- Evm_16_Qam_34: float or bool: Limits for 16QAM, coding rate 3/4
- Evm_64_Qam_12: float or bool: Limits for 64QAM, coding rate 1/2
- Evm_64_Qam_34: float or bool: Limits for 64QAM, coding rate 3/4
- Evm_64_Qam_56: float or bool: Limits for 64QAM, coding rate 5/6
- Evm_256_Qam_34: float or bool: Limits for 256QAM, coding rate 3/4
- Evm_256_Qam_56: float or bool: Limits for 256QAM, coding rate 5/6
- Evm_1024_Qam_34: float or bool: Optional setting parameter. Limits for 1024QAM, coding rate 3/4
- Evm_1024_Qam_56: float or bool: Optional setting parameter. Limits for 1024QAM, coding rate 5/6

get_cf_error() → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:VHTofdm:CFERror
value: float or bool = driver.configure.wlanMeas.multiEval.limit.modulation.
↪vhtOfdm.get_cf_error()
```

Defines and activates an upper limit for the center frequency error in 802.11ac signals.

return
center_freq_error: (float or boolean) Note that the reset value is identical for all standards.

get_evm_all() → EvmAllStruct

```
# SCPI: CONFigure:WLAN:MEASurement<instance>
↪:MEValuation:LIMit:MODulation:VHTofdm:EVMall
value: EvmAllStruct = driver.configure.wlanMeas.multiEval.limit.modulation.
↪vht0fdm.get_evm_all()
```

Defines and activates upper limits for the error vector magnitude (EVM) of 802.11ac data carriers.

return
structure: for return value, see the help for EvmAllStruct structure arguments.

get_evm_pilot() → float

```
# SCPI: CONFigure:WLAN:MEASurement<instance>
↪:MEValuation:LIMit:MODulation:VHTofdm:EVMPilot
value: float or bool = driver.configure.wlanMeas.multiEval.limit.modulation.
↪vht0fdm.get_evm_pilot()
```

Defines and activates an upper limit for the error vector magnitude (EVM) of the pilot carriers in 802.11ac signals.

return
evm_pilot: (float or boolean) No help available

get_sc_error() → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:VHTofdm:SCERror
value: float or bool = driver.configure.wlanMeas.multiEval.limit.modulation.
↪vht0fdm.get_sc_error()
```

Defines and activates an upper limit for the symbol clock error in 802.11ac signals.

return
clock_error: (float or boolean) No help available

set_cf_error(*center_freq_error: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:VHTofdm:CFERror
driver.configure.wlanMeas.multiEval.limit.modulation.vht0fdm.set_cf_
↪error(center_freq_error = 1.0)
```

Defines and activates an upper limit for the center frequency error in 802.11ac signals.

param center_freq_error
(float or boolean) Note that the reset value is identical for all standards.

set_evm_all(*value: EvmAllStruct*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<instance>
↪:MEValuation:LIMit:MODulation:VHTofdm:EVMall
structure = driver.configure.wlanMeas.multiEval.limit.modulation.vht0fdm.
↪EvmAllStruct()
structure.Evm_Br_12: float or bool = 1.0
structure.Evm_Qr_12: float or bool = 1.0
structure.Evm_Qr_34: float or bool = 1.0
```

(continues on next page)

(continued from previous page)

```
structure.Evm_16_Qam_12: float or bool = 1.0
structure.Evm_16_Qam_34: float or bool = 1.0
structure.Evm_64_Qam_12: float or bool = 1.0
structure.Evm_64_Qam_34: float or bool = 1.0
structure.Evm_64_Qam_56: float or bool = 1.0
structure.Evm_256_Qam_34: float or bool = 1.0
structure.Evm_256_Qam_56: float or bool = 1.0
structure.Evm_1024_Qam_34: float or bool = 1.0
structure.Evm_1024_Qam_56: float or bool = 1.0
driver.configure.wlanMeas.multiEval.limit.modulation.vht0fdm.set_evm_all(value␣
↪= structure)
```

Defines and activates upper limits for the error vector magnitude (EVM) of 802.11ac data carriers.

param value
see the help for EvmAllStruct structure arguments.

set_evm_pilot(*evm_pilot: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<instance>
↪:MEValuation:LIMit:MODulation:VHTofdm:EVMPilot
driver.configure.wlanMeas.multiEval.limit.modulation.vht0fdm.set_evm_pilot(evm_
↪pilot = 1.0)
```

Defines and activates an upper limit for the error vector magnitude (EVM) of the pilot carriers in 802.11ac signals.

param evm_pilot
(float or boolean) No help available

set_sc_error(*clock_error: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:VHTofdm:SCERror
driver.configure.wlanMeas.multiEval.limit.modulation.vht0fdm.set_sc_error(clock_
↪error = 1.0)
```

Defines and activates an upper limit for the symbol clock error in 802.11ac signals.

param clock_error
(float or boolean) No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.modulation.vht0fdm.clone()
```

Subgroups

6.1.1.3.3.19 IqOffset

class IqOffsetCls

IqOffset commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.modulation.vhtOfdm.iqOffset.clone()
```

Subgroups

6.1.1.3.3.20 Bw<BandwidthE>

RepCap Settings

```
# Range: Bw5 .. Bw8080
rc = driver.configure.wlanMeas.multiEval.limit.modulation.vhtOfdm.iqOffset.bw.repcap_
↪bandwidthE_get()
driver.configure.wlanMeas.multiEval.limit.modulation.vhtOfdm.iqOffset.bw.repcap_
↪bandwidthE_set(repcap.BandwidthE.Bw5)
```

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MODulation:VHTofdm:IQOFfset:BW<BW>
```

class BwCls

Bw commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: BandwidthE, default value after init: BandwidthE.Bw5

class BwStruct

Response structure. Fields:

- Offset_Value_Rel: float or bool: Relative limit
- Offset_Value_Abs: float or bool: Absolute limit, not relevant for CMP180

get(*bandwidthE=BandwidthE.Default*) → BwStruct

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:VHTofdm:IQOFfset:BW<BW>
value: BwStruct = driver.configure.wlanMeas.multiEval.limit.modulation.vhtOfdm.
↪iqOffset.bw.get(bandwidthE = repcap.BandwidthE.Default)
```

Defines and activates upper limits for the I/Q origin offset, for 802.11ac and channel bandwidth <BW>.

param bandwidthE
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

return
structure: for return value, see the help for BwStruct structure arguments.

set(*offset_value_rel: float*, *offset_value_abs: float = None*, *bandwidthE=BandwidthE.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:MODulation:VHTofdm:IQOFfset:BW<BW>
driver.configure.wlanMeas.multiEval.limit.modulation.vht0fdm.iqOffset.bw.
↪set(offset_value_rel = 1.0, offset_value_abs = 1.0, bandwidthE = repcap.
↪BandwidthE.Default)
```

Defines and activates upper limits for the I/Q origin offset, for 802.11ac and channel bandwidth <BW>.

param offset_value_rel
(float or boolean) Relative limit

param offset_value_abs
(float or boolean) Absolute limit, not relevant for CMP180

param bandwidthE
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.modulation.vht0fdm.iqOffset.bw.clone()
```

6.1.1.3.3.21 PowerVsTime

SCPI Commands :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:PVTime:REDGe
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:PVTime:FEDGe
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:PVTime:TERRor
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:PVTime:TEDistrib
```

class PowerVsTimeCls

PowerVsTime commands group definition. 4 total commands, 0 Subgroups, 4 group commands

get_falling_edge() → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:PVTime:FEDGe
value: float or bool = driver.configure.wlanMeas.multiEval.limit.powerVsTime.
↪get_falling_edge()
```

Sets the upper limit for the fall time (transmit power-down ramp) of a DSSS signal.

return
falling_limit: (float or boolean) No help available

get_rising_edge() → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:PVTime:REDGe
value: float or bool = driver.configure.wlanMeas.multiEval.limit.powerVsTime.
↪get_rising_edge()
```

Sets the upper limit for the rise time (transmit power-on ramp) of a DSSS signal.

return
rising_limit: (float or boolean) No help available

get_te_distribution() → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:PVTime:TEDistrib
value: float or bool = driver.configure.wlanMeas.multiEval.limit.powerVsTime.
↪get_te_distribution()
```

Configure the limit of timing error distribution for power vs time measurements for OFDM standards. Exceeding the limit has no impact on the stop On Limit Failure condition or out of tolerance counter.

return
te_percentage: (float or boolean) No help available

get_terror() → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:PVTime:TERRor
value: float or bool = driver.configure.wlanMeas.multiEval.limit.powerVsTime.
↪get_terror()
```

Sets the upper limit for timing error for OFDM standards.

return
timing_error: (float or boolean) No help available

set_falling_edge(*falling_limit: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:PVTime:FEDGe
driver.configure.wlanMeas.multiEval.limit.powerVsTime.set_falling_edge(falling_
↪limit = 1.0)
```

Sets the upper limit for the fall time (transmit power-down ramp) of a DSSS signal.

param falling_limit
(float or boolean) No help available

set_rising_edge(*rising_limit: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:PVTime:REDGe
driver.configure.wlanMeas.multiEval.limit.powerVsTime.set_rising_edge(rising_
↪limit = 1.0)
```

Sets the upper limit for the rise time (transmit power-on ramp) of a DSSS signal.

param rising_limit
(float or boolean) No help available

set_te_distribution(*te_percentage: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:PVTime:TEDistrib
driver.configure.wlanMeas.multiEval.limit.powerVsTime.set_te_distribution(te_
↪percentage = 1.0)
```

Configure the limit of timing error distribution for power vs time measurements for OFDM standards. Exceeding the limit has no impact on the stop On Limit Failure condition or out of tolerance counter.

param te_percentage
(float or boolean) No help available

set_terror(*timing_error: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:PVTime:TERRor
driver.configure.wlanMeas.multiEval.limit.powerVsTime.set_terror(timing_error =␣
↪1.0)
```

Sets the upper limit for timing error for OFDM standards.

param timing_error
(float or boolean) No help available

6.1.1.3.3.22 SpectrFlatness

class SpectrFlatnessCls

SpectrFlatness commands group definition. 18 total commands, 6 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.spectrFlatness.clone()
```

Subgroups

6.1.1.3.3.23 EhtOfdm

class EhtOfdmCls

EhtOfdm commands group definition. 3 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.spectrFlatness.ehtOfdm.clone()
```

Subgroups

6.1.1.3.3.24 Bw<BandwidthF>

RepCap Settings

```
# Range: Bw20 .. Bw320
rc = driver.configure.wlanMeas.multiEval.limit.spectrFlatness.ehtOfdm.bw.repcap_
↪bandwidthF_get()
driver.configure.wlanMeas.multiEval.limit.spectrFlatness.ehtOfdm.bw.repcap_bandwidthF_
↪set(repcap.BandwidthF.Bw20)
```

class BwCls

Bw commands group definition. 3 total commands, 3 Subgroups, 0 group commands Repeated Capability: BandwidthF, default value after init: BandwidthF.Bw20

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.spectrFlatness.ehtOfdm.bw.clone()
```

Subgroups

6.1.1.3.3.25 Enable

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFLatness:EHTofdm:BW<bandwidth>
↪:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthF=BandwidthF.Default*) → bool

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:SFLatness:EHTofdm:BW<bandwidth>:ENABle
value: bool = driver.configure.wlanMeas.multiEval.limit.spectrFlatness.ehtOfdm.
↪bw.enable.get(bandwidthF = repcap.BandwidthF.Default)
```

Enables or disables the spectrum flatness limit check for 802.11be signals with the specified <bandwidth>.

param bandwidthF
optional repeated capability selector. Default value: Bw20 (settable in the interface 'Bw')

return
enable: No help available

set(*enable: bool*, *bandwidthF=BandwidthF.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:SFLatness:EHTofdm:BW<bandwidth>:ENABle
driver.configure.wlanMeas.multiEval.limit.spectrFlatness.ehtOfdm.bw.enable.
↪set(enable = False, bandwidthF = repcap.BandwidthF.Default)
```

Enables or disables the spectrum flatness limit check for 802.11be signals with the specified <bandwidth>.

param enable
No help available

param bandwidthF
optional repeated capability selector. Default value: Bw20 (settable in the interface 'Bw')

6.1.1.3.3.26 Lower

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFLatness:EHTofdm:BW<bandwidth>
↪:LOWer
```

class LowerCls

Lower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class LowerStruct

Response structure. Fields:

- Center: float: No parameter help available
- Side: float: No parameter help available

get(*bandwidthF=BandwidthF.Default*) → LowerStruct

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:SFLatness:EHTofdm:BW<bandwidth>:LOWer
value: LowerStruct = driver.configure.wlanMeas.multiEval.limit.spectrFlatness.
↪ehtOfdm.bw.lower.get(bandwidthF = repcap.BandwidthF.Default)
```

Defines the lower limits for the spectrum flatness of the center subcarriers and the side subcarriers for 802.11be signals with the specified <bandwidth>. The lower limits must be smaller than the upper limit.

param bandwidthF
optional repeated capability selector. Default value: Bw20 (settable in the interface 'Bw')

return
structure: for return value, see the help for LowerStruct structure arguments.

set(*center: float*, *side: float*, *bandwidthF=BandwidthF.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:SFLatness:EHTofdm:BW<bandwidth>:LOWer
driver.configure.wlanMeas.multiEval.limit.spectrFlatness.ehtOfdm.bw.lower.
↪set(center = 1.0, side = 1.0, bandwidthF = repcap.BandwidthF.Default)
```

Defines the lower limits for the spectrum flatness of the center subcarriers and the side subcarriers for 802.11be signals with the specified <bandwidth>. The lower limits must be smaller than the upper limit.

param center
No help available

param side
No help available

param bandwidthF
optional repeated capability selector. Default value: Bw20 (settable in the interface 'Bw')

6.1.1.3.3.27 Upper

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFLatness:EHTofdm:BW<bandwidth>
↪:UPPer
```

class UpperCls

Upper commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthF=BandwidthF.Default*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:SFLatness:EHTofdm:BW<bandwidth>:UPPer
value: float = driver.configure.wlanMeas.multiEval.limit.spectrFlatness.ehtOfdm.
↪bw.upper.get(bandwidthF = repcap.BandwidthF.Default)
```

Defines the upper limit for the spectrum flatness of 802.11be signals with the specified <bandwidth>. The upper limit must be larger than the lower limits.

param bandwidthF
optional repeated capability selector. Default value: Bw20 (settable in the interface 'Bw')

return
upper: No help available

set(*upper: float*, *bandwidthF=BandwidthF.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:SFLatness:EHTofdm:BW<bandwidth>:UPPer
driver.configure.wlanMeas.multiEval.limit.spectrFlatness.ehtOfdm.bw.upper.
↪set(upper = 1.0, bandwidthF = repcap.BandwidthF.Default)
```

Defines the upper limit for the spectrum flatness of 802.11be signals with the specified <bandwidth>. The upper limit must be larger than the lower limits.

param upper
No help available

param bandwidthF
optional repeated capability selector. Default value: Bw20 (settable in the interface 'Bw')

6.1.1.3.3.28 HeOfdm

class HeOfdmCls

HeOfdm commands group definition. 3 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.spectrFlatness.heOfdm.clone()
```

Subgroups

6.1.1.3.3.29 Bw<BandwidthD>

RepCap Settings

```
# Range: Bw20 .. Bw8080
rc = driver.configure.wlanMeas.multiEval.limit.spectrFlatness.heOfdm.bw.repcap_
↪bandwidthD_get()
driver.configure.wlanMeas.multiEval.limit.spectrFlatness.heOfdm.bw.repcap_bandwidthD_
↪set(repcap.BandwidthD.Bw20)
```

class BwCls

Bw commands group definition. 3 total commands, 3 Subgroups, 0 group commands Repeated Capability: BandwidthD, default value after init: BandwidthD.Bw20

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.spectrFlatness.heOfdm.bw.clone()
```

Subgroups

6.1.1.3.3.30 Enable

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFLatness:HEOFdm:BW<bandwidth>
↪:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthD=BandwidthD.Default*) → bool

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:SFLatness:HEOFdm:BW<bandwidth>:ENABle
value: bool = driver.configure.wlanMeas.multiEval.limit.spectrFlatness.heOfdm.
↪bw.enable.get(bandwidthD = repcap.BandwidthD.Default)
```

Enables or disables the spectrum flatness limit check for 802.11ax signals with the specified <bandwidth>.

param bandwidthD
optional repeated capability selector. Default value: Bw20 (settable in the interface 'Bw')

return
enable: No help available

set(*enable: bool*, *bandwidthD=BandwidthD.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:SFLatness:HEOFdm:BW<bandwidth>:ENABle
driver.configure.wlanMeas.multiEval.limit.spectrFlatness.heOfdm.bw.enable.
↪set(enable = False, bandwidthD = repcap.BandwidthD.Default)
```

Enables or disables the spectrum flatness limit check for 802.11ax signals with the specified <bandwidth>.

param enable
No help available

param bandwidthD
optional repeated capability selector. Default value: Bw20 (settable in the interface 'Bw')

6.1.1.3.3.31 Lower

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFLatness:HEOFdm:BW<bandwidth>
↪:LOWer
```

class LowerCls

Lower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class LowerStruct

Response structure. Fields:

- Center: float: No parameter help available
- Side: float: No parameter help available

get(*bandwidthD=BandwidthD.Default*) → LowerStruct

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:SFLatness:HEOFdm:BW<bandwidth>:LOWer
value: LowerStruct = driver.configure.wlanMeas.multiEval.limit.spectrFlatness.
↪heOfdm.bw.lower.get(bandwidthD = repcap.BandwidthD.Default)
```

Defines the lower limits for the spectrum flatness of the center subcarriers and the side subcarriers for 802.11ax signals with the specified <bandwidth>. The lower limits must be smaller than the upper limit.

param bandwidthD
optional repeated capability selector. Default value: Bw20 (settable in the interface 'Bw')

return
structure: for return value, see the help for LowerStruct structure arguments.

`set`(*center: float*, *side: float*, *bandwidthD=BandwidthD.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:SFLatness:HEOFdm:BW<bandwidth>:LOWer
driver.configure.wlanMeas.multiEval.limit.spectrFlatness.heOfdm.bw.lower.
↪set(center = 1.0, side = 1.0, bandwidthD = repcap.BandwidthD.Default)
```

Defines the lower limits for the spectrum flatness of the center subcarriers and the side subcarriers for 802.11ax signals with the specified <bandwidth>. The lower limits must be smaller than the upper limit.

param center
No help available

param side
No help available

param bandwidthD
optional repeated capability selector. Default value: Bw20 (settable in the interface 'Bw')

6.1.1.3.3.32 Upper

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFLatness:HEOFdm:BW<bandwidth>
↪:UPPer
```

`class UpperCls`

Upper commands group definition. 1 total commands, 0 Subgroups, 1 group commands

`get`(*bandwidthD=BandwidthD.Default*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:SFLatness:HEOFdm:BW<bandwidth>:UPPer
value: float = driver.configure.wlanMeas.multiEval.limit.spectrFlatness.heOfdm.
↪bw.upper.get(bandwidthD = repcap.BandwidthD.Default)
```

Defines the upper limit for the spectrum flatness of 802.11ax signals with the specified <bandwidth>. The upper limit must be larger than the lower limits.

param bandwidthD
optional repeated capability selector. Default value: Bw20 (settable in the interface 'Bw')

return
upper: No help available

`set`(*upper: float*, *bandwidthD=BandwidthD.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:SFLatness:HEOFdm:BW<bandwidth>:UPPer
driver.configure.wlanMeas.multiEval.limit.spectrFlatness.heOfdm.bw.upper.
↪set(upper = 1.0, bandwidthD = repcap.BandwidthD.Default)
```

Defines the upper limit for the spectrum flatness of 802.11ax signals with the specified <bandwidth>. The upper limit must be larger than the lower limits.

param upper
No help available

param bandwidthD
optional repeated capability selector. Default value: Bw20 (settable in the interface 'Bw')

6.1.1.3.3.33 HtOfdm

class HtOfdmCls

HtOfdm commands group definition. 3 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.spectrFlatness.htOfdm.clone()
```

Subgroups

6.1.1.3.3.34 Bw<BandwidthC>

RepCap Settings

```
# Range: Bw5 .. Bw40
rc = driver.configure.wlanMeas.multiEval.limit.spectrFlatness.htOfdm.bw.repcap_
↪bandwidthC_get()
driver.configure.wlanMeas.multiEval.limit.spectrFlatness.htOfdm.bw.repcap_bandwidthC_
↪set(repcap.BandwidthC.Bw5)
```

class BwCls

Bw commands group definition. 3 total commands, 3 Subgroups, 0 group commands Repeated Capability: BandwidthC, default value after init: BandwidthC.Bw5

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.spectrFlatness.htOfdm.bw.clone()
```

Subgroups

6.1.1.3.3.35 Enable

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFLatness:HTOFdm:BW<bandwidth>
↪:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthC=BandwidthC.Default*) → bool

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:SFLatness:HTOFdm:BW<bandwidth>:ENABle
value: bool = driver.configure.wlanMeas.multiEval.limit.spectrFlatness.htOfdm.
↪bw.enable.get(bandwidthC = repcap.BandwidthC.Default)
```

Enables or disables the spectrum flatness limit check for 802.11n signals with the specified <bandwidth>.

param bandwidthC
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

return
enable: No help available

set(*enable: bool*, *bandwidthC=BandwidthC.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:SFLatness:HTOFdm:BW<bandwidth>:ENABle
driver.configure.wlanMeas.multiEval.limit.spectrFlatness.htOfdm.bw.enable.
↪set(enable = False, bandwidthC = repcap.BandwidthC.Default)
```

Enables or disables the spectrum flatness limit check for 802.11n signals with the specified <bandwidth>.

param enable
No help available

param bandwidthC
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

6.1.1.3.3.36 Lower

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFLatness:HTOFdm:BW<bandwidth>
↪:LOWer
```

class LowerCls

Lower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class LowerStruct

Response structure. Fields:

- Center: float: No parameter help available
- Side: float: No parameter help available

get(*bandwidthC=BandwidthC.Default*) → LowerStruct

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:SFLatness:HTOFdm:BW<bandwidth>:LOWer
value: LowerStruct = driver.configure.wlanMeas.multiEval.limit.spectrFlatness.
↪htOfdm.bw.lower.get(bandwidthC = repcap.BandwidthC.Default)
```

Defines lower limits for the spectrum flatness of the center subcarriers and the side subcarriers for 802.11n signals with the specified <bandwidth>. The lower limits must be smaller than the upper limit.

param bandwidthC
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

return
structure: for return value, see the help for LowerStruct structure arguments.

set(*center: float*, *side: float*, *bandwidthC=BandwidthC.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:SFLatness:HTOFdm:BW<bandwidth>:LOWer
driver.configure.wlanMeas.multiEval.limit.spectrFlatness.htOfdm.bw.lower.
↪set(center = 1.0, side = 1.0, bandwidthC = repcap.BandwidthC.Default)
```

Defines lower limits for the spectrum flatness of the center subcarriers and the side subcarriers for 802.11n signals with the specified <bandwidth>. The lower limits must be smaller than the upper limit.

param center
No help available

param side
No help available

param bandwidthC
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

6.1.1.3.3.37 Upper

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFLatness:HTOFdm:BW<bandwidth>
↪:UPPer
```

class UpperCls

Upper commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthC=BandwidthC.Default*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:SFLatness:HTOFdm:BW<bandwidth>:UPPer
value: float = driver.configure.wlanMeas.multiEval.limit.spectrFlatness.htOfdm.
↪bw.upper.get(bandwidthC = repcap.BandwidthC.Default)
```

Defines an upper limit for the spectrum flatness of 802.11n signals with the specified <bandwidth>. The upper limit must be larger than the lower limits.

param bandwidthC
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

return
upper: No help available

set(*upper: float*, *bandwidthC=BandwidthC.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:SFLatness:HTOFdm:BW<bandwidth>:UPPer
driver.configure.wlanMeas.multiEval.limit.spectrFlatness.htOfdm.bw.upper.
↪set(upper = 1.0, bandwidthC = repcap.BandwidthC.Default)
```

Defines an upper limit for the spectrum flatness of 802.11n signals with the specified <bandwidth>. The upper limit must be larger than the lower limits.

param upper
No help available

param bandwidthC
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

6.1.1.3.3.38 Lofdm

SCPI Commands :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFLatness:LOFDm:ENABle
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFLatness:LOFDm:UPPer
```

class LofdmCls

Lofdm commands group definition. 3 total commands, 1 Subgroups, 2 group commands

get_enable() → bool

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:SFLatness:LOFDm:ENABle
value: bool = driver.configure.wlanMeas.multiEval.limit.spectrFlatness.lofdm.
↪get_enable()
```

No command help available

return
enable: No help available

get_upper() → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:SFLatness:LOFDm:UPPer
value: float = driver.configure.wlanMeas.multiEval.limit.spectrFlatness.lofdm.
↪get_upper()
```

No command help available

return
upper: No help available

set_enable(*enable: bool*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:SFLatness:LOFDm:ENABle
driver.configure.wlanMeas.multiEval.limit.spectrFlatness.lofdm.set_
↪enable(enable = False)
```

No command help available

param enable
No help available

set_upper(*upper: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:SFLatness:LOFDm:UPPer
driver.configure.wlanMeas.multiEval.limit.spectrFlatness.lofdm.set_upper(upper␣
↪= 1.0)
```

No command help available

param upper
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.spectrFlatness.lofdm.clone()
```

Subgroups

6.1.1.3.3.39 Lower

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFLatness:LOFDm:LOWer
```

class LowerCls

Lower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class LowerStruct

Response structure. Fields:

- Center: float: No parameter help available
- Side: float: No parameter help available

get() → LowerStruct

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:SFLatness:LOFDm:LOWer
value: LowerStruct = driver.configure.wlanMeas.multiEval.limit.spectrFlatness.
↪lofdm.lower.get()
```

No command help available

return
structure: for return value, see the help for LowerStruct structure arguments.

set(*center: float*, *side: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:SFLatness:LOFDm:LOWer
driver.configure.wlanMeas.multiEval.limit.spectrFlatness.lofdm.lower.set(center␣
↪= 1.0, side = 1.0)
```

No command help available

param center
No help available

param side
No help available

6.1.1.3.3.40 Pofdm

class PofdmCls

Pofdm commands group definition. 3 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.spectrFlatness.pofdm.clone()
```

Subgroups

6.1.1.3.3.41 Bw<BandwidthB>

RepCap Settings

```
# Range: Bw5 .. Bw20
rc = driver.configure.wlanMeas.multiEval.limit.spectrFlatness.pofdm.bw.repcap_bandwidthB_
↪get()
driver.configure.wlanMeas.multiEval.limit.spectrFlatness.pofdm.bw.repcap_bandwidthB_
↪set(repcap.BandwidthB.Bw5)
```

class BwCls

Bw commands group definition. 3 total commands, 3 Subgroups, 0 group commands Repeated Capability: BandwidthB, default value after init: BandwidthB.Bw5

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.spectrFlatness.pofdm.bw.clone()
```

Subgroups

6.1.1.3.3.42 Enable

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFLatness:POFDm:BW<bandwidth>
↪:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthB=BandwidthB.Default*) → bool

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:SFLatness:POFDm:BW<bandwidth>:ENABle
value: bool = driver.configure.wlanMeas.multiEval.limit.spectrFlatness.pofdm.bw.
↪enable.get(bandwidthB = repcap.BandwidthB.Default)
```

Enables or disables the spectrum flatness limit check for 802.11p OFDM signals with the specified <bandwidth>.

param bandwidthB
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

return
enable: No help available

set(*enable: bool*, *bandwidthB=BandwidthB.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:SFLatness:POFDm:BW<bandwidth>:ENABle
driver.configure.wlanMeas.multiEval.limit.spectrFlatness.pofdm.bw.enable.
↪set(enable = False, bandwidthB = repcap.BandwidthB.Default)
```

Enables or disables the spectrum flatness limit check for 802.11p OFDM signals with the specified <bandwidth>.

param enable
No help available

param bandwidthB
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

6.1.1.3.3.43 Lower

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFLatness:POFDm:BW<bandwidth>
↪:LOWer
```

class LowerCls

Lower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class LowerStruct

Response structure. Fields:

- Center: float: No parameter help available
- Side: float: No parameter help available

get(*bandwidthB=BandwidthB.Default*) → LowerStruct

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:SFLatness:POFDm:BW<bandwidth>:LOWer
value: LowerStruct = driver.configure.wlanMeas.multiEval.limit.spectrFlatness.
↪pofdm.bw.lower.get(bandwidthB = repcap.BandwidthB.Default)
```

Defines lower limits for the spectrum flatness of the center subcarriers and the side subcarriers of 802.11p OFDM signals with the specified <bandwidth>. The lower limits must be smaller than the upper limit.

param bandwidthB
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

return
structure: for return value, see the help for LowerStruct structure arguments.

set(*center: float*, *side: float*, *bandwidthB=BandwidthB.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:SFLatness:POFDm:BW<bandwidth>:LOWer
driver.configure.wlanMeas.multiEval.limit.spectrFlatness.pofdm.bw.lower.
↪set(center = 1.0, side = 1.0, bandwidthB = repcap.BandwidthB.Default)
```

Defines lower limits for the spectrum flatness of the center subcarriers and the side subcarriers of 802.11p OFDM signals with the specified <bandwidth>. The lower limits must be smaller than the upper limit.

param center
No help available

param side
No help available

param bandwidthB
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

6.1.1.3.3.44 Upper

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFLatness:POFDm:BW<bandwidth>
↪:UPPer
```

class UpperCls

Upper commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthB=BandwidthB.Default*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:SFLatness:POFDm:BW<bandwidth>:UPPer
value: float = driver.configure.wlanMeas.multiEval.limit.spectrFlatness.pofdm.
↪bw.upper.get(bandwidthB = repcap.BandwidthB.Default)
```

Defines an upper limit for the spectrum flatness of 802.11p OFDM signals with the specified <bandwidth>. The upper limit must be larger than the lower limit.

param bandwidthB
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

return
upper: No help available

set(*upper: float*, *bandwidthB=BandwidthB.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:SFLatness:POFDm:BW<bandwidth>:UPPer
driver.configure.wlanMeas.multiEval.limit.spectrFlatness.pofdm.bw.upper.
↪set(upper = 1.0, bandwidthB = repcap.BandwidthB.Default)
```

Defines an upper limit for the spectrum flatness of 802.11p OFDM signals with the specified <bandwidth>. The upper limit must be larger than the lower limit.

param upper
No help available

param bandwidthB
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

6.1.1.3.3.45 VhtOfdm

class VhtOfdmCls

VhtOfdm commands group definition. 3 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.spectrFlatness.vhtOfdm.clone()
```

Subgroups

6.1.1.3.3.46 Bw<BandwidthE>

RepCap Settings

```
# Range: Bw5 .. Bw8080
rc = driver.configure.wlanMeas.multiEval.limit.spectrFlatness.vhtOfdm.bw.repcap_
↪bandwidthE_get()
driver.configure.wlanMeas.multiEval.limit.spectrFlatness.vhtOfdm.bw.repcap_bandwidthE_
↪set(repcap.BandwidthE.Bw5)
```

class BwCls

Bw commands group definition. 3 total commands, 3 Subgroups, 0 group commands Repeated Capability: BandwidthE, default value after init: BandwidthE.Bw5

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.spectrFlatness.vht0fdm.bw.clone()
```

Subgroups

6.1.1.3.3.47 Enable

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFLatness:VHTofdm:BW<bandwidth>
↪:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthE=BandwidthE.Default*) → bool

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:SFLatness:VHTofdm:BW<bandwidth>:ENABle
value: bool = driver.configure.wlanMeas.multiEval.limit.spectrFlatness.vht0fdm.
↪bw.enable.get(bandwidthE = repcap.BandwidthE.Default)
```

Enables or disables the spectrum flatness limit check for 802.11ac signals with the specified <bandwidth>.

param bandwidthE
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

return
enable: No help available

set(*enable: bool*, *bandwidthE=BandwidthE.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:SFLatness:VHTofdm:BW<bandwidth>:ENABle
driver.configure.wlanMeas.multiEval.limit.spectrFlatness.vht0fdm.bw.enable.
↪set(enable = False, bandwidthE = repcap.BandwidthE.Default)
```

Enables or disables the spectrum flatness limit check for 802.11ac signals with the specified <bandwidth>.

param enable
No help available

param bandwidthE
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

6.1.1.3.3.48 Lower

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFLatness:VHTofdm:BW<bandwidth>
↪:LOWer
```

class LowerCls

Lower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class LowerStruct

Response structure. Fields:

- Center: float: No parameter help available
- Side: float: No parameter help available

get(*bandwidthE=BandwidthE.Default*) → LowerStruct

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:SFLatness:VHTofdm:BW<bandwidth>:LOWer
value: LowerStruct = driver.configure.wlanMeas.multiEval.limit.spectrFlatness.
↪vhtOfdm.bw.lower.get(bandwidthE = repcap.BandwidthE.Default)
```

Defines lower limits for the spectrum flatness of the center subcarriers and the side subcarriers for 802.11ac signals with the specified <bandwidth>. The lower limits must be smaller than the upper limit.

param bandwidthE
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

return
structure: for return value, see the help for LowerStruct structure arguments.

set(*center: float*, *side: float*, *bandwidthE=BandwidthE.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:SFLatness:VHTofdm:BW<bandwidth>:LOWer
driver.configure.wlanMeas.multiEval.limit.spectrFlatness.vhtOfdm.bw.lower.
↪set(center = 1.0, side = 1.0, bandwidthE = repcap.BandwidthE.Default)
```

Defines lower limits for the spectrum flatness of the center subcarriers and the side subcarriers for 802.11ac signals with the specified <bandwidth>. The lower limits must be smaller than the upper limit.

param center
No help available

param side
No help available

param bandwidthE
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

6.1.1.3.3.49 Upper

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFLatness:VHTofdm:BW<bandwidth>
↪:UPPer
```

class UpperCls

Upper commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthE=BandwidthE.Default*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:SFLatness:VHTofdm:BW<bandwidth>:UPPer
value: float = driver.configure.wlanMeas.multiEval.limit.spectrFlatness.vht0fdm.
↪bw.upper.get(bandwidthE = repcap.BandwidthE.Default)
```

Defines an upper limit for the spectrum flatness of 802.11ac signals with the specified <bandwidth>. The upper limit must be larger than the lower limits.

param bandwidthE
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

return
upper: No help available

set(*upper: float*, *bandwidthE=BandwidthE.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:SFLatness:VHTofdm:BW<bandwidth>:UPPer
driver.configure.wlanMeas.multiEval.limit.spectrFlatness.vht0fdm.bw.upper.
↪set(upper = 1.0, bandwidthE = repcap.BandwidthE.Default)
```

Defines an upper limit for the spectrum flatness of 802.11ac signals with the specified <bandwidth>. The upper limit must be larger than the lower limits.

param upper
No help available

param bandwidthE
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

6.1.1.3.3.50 TsMask

class TsMaskCls

TsMask commands group definition. 54 total commands, 7 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.tsMask.clone()
```

Subgroups

6.1.1.3.3.51 Dsss

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:DSSS:ENABle
```

class DsssCls

Dsss commands group definition. 3 total commands, 1 Subgroups, 1 group commands

get_enable() → bool

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:TSMask:DSSS:ENABle
value: bool = driver.configure.wlanMeas.multiEval.limit.tsMask.dsss.get_enable()
```

Activates or deactivates the transmit spectrum mask (transmission scheme DSSS) , i.e. the limit check.

return
spe_lim_enable: No help available

set_enable(*spe_lim_enable: bool*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:TSMask:DSSS:ENABle
driver.configure.wlanMeas.multiEval.limit.tsMask.dsss.set_enable(spe_lim_enable␣
↪= False)
```

Activates or deactivates the transmit spectrum mask (transmission scheme DSSS) , i.e. the limit check.

param spe_lim_enable
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.tsMask.dsss.clone()
```

Subgroups

6.1.1.3.3.52 Y

SCPI Commands :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:DSSS:Y:AB
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:DSSS:Y:CD
```

class YCls

Y commands group definition. 2 total commands, 0 Subgroups, 2 group commands

get_ab() → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:DSSS:Y:AB
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.dsss.y.get_ab()
```

Defines the power level of the horizontal spectrum mask line connecting point A and B, see 'Transmit spectrum mask DSSS'.

return
yrel_level_ab: No help available

get_cd() → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:DSSS:Y:CD
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.dsss.y.get_cd()
```

Defines the power level of the horizontal spectrum mask line connecting point C and D, see 'Transmit spectrum mask DSSS'.

return
yrel_level_cd: No help available

set_ab(*yrel_level_ab: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:DSSS:Y:AB
driver.configure.wlanMeas.multiEval.limit.tsMask.dsss.y.set_ab(yrel_level_ab =␣
↪1.0)
```

Defines the power level of the horizontal spectrum mask line connecting point A and B, see 'Transmit spectrum mask DSSS'.

param yrel_level_ab
No help available

set_cd(*yrel_level_cd: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:DSSS:Y:CD
driver.configure.wlanMeas.multiEval.limit.tsMask.dsss.y.set_cd(yrel_level_cd =␣
↪1.0)
```

Defines the power level of the horizontal spectrum mask line connecting point C and D, see 'Transmit spectrum mask DSSS'.

param yrel_level_cd
No help available

6.1.1.3.3.53 EhtOfdm

class EhtOfdmCls

EhtOfdm commands group definition. 6 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.tsMask.ehtOfdm.clone()
```

Subgroups

6.1.1.3.3.54 Bw<BandwidthF>

RepCap Settings

```
# Range: Bw20 .. Bw320
rc = driver.configure.wlanMeas.multiEval.limit.tsMask.ehtOfdm.bw.repcap_bandwidthF_get()
driver.configure.wlanMeas.multiEval.limit.tsMask.ehtOfdm.bw.repcap_bandwidthF_set(repcap.
↪BandwidthF.Bw20)
```

class BwCls

Bw commands group definition. 6 total commands, 3 Subgroups, 0 group commands Repeated Capability: BandwidthF, default value after init: BandwidthF.Bw20

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.tsMask.ehtOfdm.bw.clone()
```

Subgroups

6.1.1.3.3.55 AbsLimit

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:EHTofdm:BW<bandwidth>
↪:ABSLimit
```

class AbsLimitCls

AbsLimit commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthF=BandwidthF.Default*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:EHTofdm:BW
↪<bandwidth>:ABSLimit
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.ehtOfdm.bw.
↪absLimit.get(bandwidthF = repcap.BandwidthF.Default)
```

No command help available

param bandwidthF
optional repeated capability selector. Default value: Bw20 (settable in the interface 'Bw')

return
tsm_lim_abs: No help available

set(*tsm_lim_abs: float*, *bandwidthF=BandwidthF.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:EHTofdm:BW
↪<bandwidth>:ABSLimit
driver.configure.wlanMeas.multiEval.limit.tsMask.ehtOfdm.bw.absLimit.set(tsm_
↪lim_abs = 1.0, bandwidthF = repcap.BandwidthF.Default)
```

No command help available

param tsm_lim_abs
No help available

param bandwidthF
optional repeated capability selector. Default value: Bw20 (settable in the interface 'Bw')

6.1.1.3.3.56 Enable

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:EHTofdm:BW<bandwidth>
↪:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthF=BandwidthF.Default*) → bool

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:EHTofdm:BW
↪<bandwidth>:ENABle
value: bool = driver.configure.wlanMeas.multiEval.limit.tsMask.ehtOfdm.bw.
↪enable.get(bandwidthF = repcap.BandwidthF.Default)
```

Enables or disables the transmit spectrum mask for 802.11be signals with the specified <bandwidth>, i.e. activates or deactivates the corresponding limit checks.

param bandwidthF
optional repeated capability selector. Default value: Bw20 (settable in the interface 'Bw')

return
tsm_lim_enable: No help available

set(*tsm_lim_enable: bool*, *bandwidthF=BandwidthF.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:EHTofdm:BW
↪<bandwidth>:ENABle
driver.configure.wlanMeas.multiEval.limit.tsMask.ehtOfdm.bw.enable.set(tsm_lim_
↪enable = False, bandwidthF = repcap.BandwidthF.Default)
```

Enables or disables the transmit spectrum mask for 802.11be signals with the specified <bandwidth>, i.e. activates or deactivates the corresponding limit checks.

param tsm_lim_enable
No help available

param bandwidthF
optional repeated capability selector. Default value: Bw20 (settable in the interface 'Bw')

6.1.1.3.3.57 Y

class YCls

Y commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.tsMask.ehtOfdm.bw.y.clone()
```

Subgroups

6.1.1.3.3.58 A

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:EHTofdm:BW<bandwidth>:Y:A
```

class ACls

A commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthF=BandwidthF.Default*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:EHTofdm:BW
↪<bandwidth>:Y:A
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.ehtOfdm.bw.y.a.
↪get(bandwidthF = repcap.BandwidthF.Default)
```

Defines the relative spectral density limit for point A (frequency offset: 2*bandwidth) on the transmit spectrum mask for 802.11be signals with the specified <bandwidth>. See 'Transmit spectrum mask OFDM, default masks' for background information.

param bandwidthF
optional repeated capability selector. Default value: Bw20 (settable in the interface 'Bw')

return
tsm_lim_yrel_lev_a: No help available

set(*tsm_lim_yrel_lev_a: float*, *bandwidthF=BandwidthF.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:EHTofdm:BW
↪<bandwidth>:Y:A
driver.configure.wlanMeas.multiEval.limit.tsMask.ehtOfdm.bw.y.a.set(tsm_lim_
↪yrel_lev_a = 1.0, bandwidthF = repcap.BandwidthF.Default)
```

Defines the relative spectral density limit for point A (frequency offset: 2*bandwidth) on the transmit spectrum mask for 802.11be signals with the specified <bandwidth>. See 'Transmit spectrum mask OFDM, default masks' for background information.

param tsm_lim_yrel_lev_a
No help available

param bandwidthF
optional repeated capability selector. Default value: Bw20 (settable in the interface 'Bw')

6.1.1.3.3.59 B

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:EHTofdm:BW<bandwidth>:Y:B
```

class BCls

B commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthF=BandwidthF.Default*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:EHTofdm:BW
↪<bandwidth>:Y:B
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.ehtOfdm.bw.y.b.
↪get(bandwidthF = repcap.BandwidthF.Default)
```

Defines the relative spectral density limit for point B (frequency offset: 3/2*bandwidth) on the transmit spectrum mask for 802.11be signals with the specified <bandwidth>. See 'Transmit spectrum mask OFDM, default masks' for background information.

param bandwidthF
optional repeated capability selector. Default value: Bw20 (settable in the interface 'Bw')

return
tsm_lim_yrel_lev_b: No help available

set(*tsm_lim_yrel_lev_b: float*, *bandwidthF=BandwidthF.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:EHTofdm:BW
↪<bandwidth>:Y:B
driver.configure.wlanMeas.multiEval.limit.tsMask.ehtOfdm.bw.y.b.set(tsm_lim_
↪yrel_lev_b = 1.0, bandwidthF = repcap.BandwidthF.Default)
```

Defines the relative spectral density limit for point B (frequency offset: 3/2*bandwidth) on the transmit spectrum mask for 802.11be signals with the specified <bandwidth>. See 'Transmit spectrum mask OFDM, default masks' for background information.

param tsm_lim_yrel_lev_b
No help available

param bandwidthF
optional repeated capability selector. Default value: Bw20 (settable in the interface 'Bw')

6.1.1.3.3.60 C

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:EHTofdm:BW<bandwidth>:Y:C
```

class CCls

C commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthF=BandwidthF.Default*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:EHTofdm:BW
↪<bandwidth>:Y:C
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.ehtOfdm.bw.y.c.
↪get(bandwidthF = repcap.BandwidthF.Default)
```

Defines the relative spectral density limit for point C (frequency offset: 1*bandwidth) on the transmit spectrum mask for 802.11be signals with the specified <bandwidth>. See 'Transmit spectrum mask OFDM, default masks' for background information.

param bandwidthF
optional repeated capability selector. Default value: Bw20 (settable in the interface 'Bw')

return
tsm_lim_yrel_lev_c: No help available

set(*tsm_lim_yrel_lev_c: float*, *bandwidthF=BandwidthF.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:EHTofdm:BW
↪<bandwidth>:Y:C
driver.configure.wlanMeas.multiEval.limit.tsMask.ehtOfdm.bw.y.c.set(tsm_lim_
↪yrel_lev_c = 1.0, bandwidthF = repcap.BandwidthF.Default)
```

Defines the relative spectral density limit for point C (frequency offset: 1*bandwidth) on the transmit spectrum mask for 802.11be signals with the specified <bandwidth>. See 'Transmit spectrum mask OFDM, default masks' for background information.

param tsm_lim_yrel_lev_c
No help available

param bandwidthF
optional repeated capability selector. Default value: Bw20 (settable in the interface 'Bw')

6.1.1.3.3.61 D

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:EHTofdm:BW<bandwidth>:Y:D
```

class DCls

D commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthF=BandwidthF.Default*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:EHTofdm:BW
↪<bandwidth>:Y:D
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.ehtOfdm.bw.y.d.
↪get(bandwidthF = repcap.BandwidthF.Default)
```

Defines the relative spectral density limit for point D (center frequency offset: 1/2*bandwidth + 1 MHz) on the transmit spectrum mask for 802.11be signals with the specified <bandwidth>. See 'Transmit spectrum mask OFDM, default masks' for background information.

param bandwidthF
optional repeated capability selector. Default value: Bw20 (settable in the interface 'Bw')

return
tsm_lim_yrel_lev_d: No help available

set(*tsm_lim_yrel_lev_d: float*, *bandwidthF=BandwidthF.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:EHTofdm:BW
↪<bandwidth>:Y:D
driver.configure.wlanMeas.multiEval.limit.tsMask.ehtOfdm.bw.y.d.set(tsm_lim_
↪yrel_lev_d = 1.0, bandwidthF = repcap.BandwidthF.Default)
```

Defines the relative spectral density limit for point D (center frequency offset: 1/2*bandwidth + 1 MHz) on the transmit spectrum mask for 802.11be signals with the specified <bandwidth>. See 'Transmit spectrum mask OFDM, default masks' for background information.

param tsm_lim_yrel_lev_d
No help available

param bandwidthF
optional repeated capability selector. Default value: Bw20 (settable in the interface 'Bw')

6.1.1.3.3.62 HeOfdm

class HeOfdmCls

HeOfdm commands group definition. 6 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.tsMask.heOfdm.clone()
```

Subgroups

6.1.1.3.3.63 Bw<BandwidthD>

RepCap Settings

```
# Range: Bw20 .. Bw8080
rc = driver.configure.wlanMeas.multiEval.limit.tsMask.heOfdm.bw.repcap_bandwidthD_get()
driver.configure.wlanMeas.multiEval.limit.tsMask.heOfdm.bw.repcap_bandwidthD_set(repcap.
↪BandwidthD.Bw20)
```

class BwCls

Bw commands group definition. 6 total commands, 3 Subgroups, 0 group commands Repeated Capability: BandwidthD, default value after init: BandwidthD.Bw20

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.tsMask.heOfdm.bw.clone()
```

Subgroups

6.1.1.3.3.64 AbsLimit

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HEOFdm:BW<bandwidth>
↪:ABSLimit
```

class AbsLimitCls

AbsLimit commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthD=BandwidthD.Default*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HEOFdm:BW
↪<bandwidth>:ABSLimit
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.heOfdm.bw.
↪absLimit.get(bandwidthD = repcap.BandwidthD.Default)
```

Defines the absolute power limit for 802.11ax signals with the specified <bandwidth>. See 'Transmit spectrum mask OFDM, absolute limits' for background information.

param bandwidthD
optional repeated capability selector. Default value: Bw20 (settable in the interface 'Bw')

return
tsm_lim_abs: Limit value applies to frequency offsets greater than 3/2*bandwidth, measured at 25 kHz RBW.

set(*tsm_lim_abs: float*, *bandwidthD=BandwidthD.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HEOFdm:BW
↪<bandwidth>:ABSLimit
driver.configure.wlanMeas.multiEval.limit.tsMask.heOfdm.bw.absLimit.set(tsm_lim_
↪abs = 1.0, bandwidthD = repcap.BandwidthD.Default)
```

Defines the absolute power limit for 802.11ax signals with the specified <bandwidth>. See 'Transmit spectrum mask OFDM, absolute limits' for background information.

param tsm_lim_abs
Limit value applies to frequency offsets greater than 3/2*bandwidth, measured at 25 kHz RBW.

param bandwidthD
optional repeated capability selector. Default value: Bw20 (settable in the interface 'Bw')

6.1.1.3.3.65 Enable

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HEOFdm:BW<bandwidth>:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthD=BandwidthD.Default*) → bool

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HEOFdm:BW
↪<bandwidth>:ENABle
value: bool = driver.configure.wlanMeas.multiEval.limit.tsMask.heOfdm.bw.enable.
↪get(bandwidthD = repcap.BandwidthD.Default)
```

Enables or disables the transmit spectrum mask for 802.11ax signals with the specified <bandwidth>, i.e. activates or deactivates the corresponding limit checks.

param bandwidthD
optional repeated capability selector. Default value: Bw20 (settable in the interface 'Bw')

return
tsm_lim_enable: No help available

set(*tsm_lim_enable: bool*, *bandwidthD=BandwidthD.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HEOFdm:BW
↪<bandwidth>:ENABle
driver.configure.wlanMeas.multiEval.limit.tsMask.heOfdm.bw.enable.set(tsm_lim_
↪enable = False, bandwidthD = repcap.BandwidthD.Default)
```

Enables or disables the transmit spectrum mask for 802.11ax signals with the specified <bandwidth>, i.e. activates or deactivates the corresponding limit checks.

param tsm_lim_enable
No help available

param bandwidthD
optional repeated capability selector. Default value: Bw20 (settable in the interface 'Bw')

6.1.1.3.3.66 Y

class YCls

Y commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.tsMask.heOfdm.bw.y.clone()
```

Subgroups

6.1.1.3.3.67 A

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HEOFdm:BW<bandwidth>:Y:A
```

class ACls

A commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthD=BandwidthD.Default*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HEOFdm:BW
↪<bandwidth>:Y:A
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.heOfdm.bw.y.a.
↪get(bandwidthD = repcap.BandwidthD.Default)
```

Defines the relative spectral density limit for point A (frequency offset: 2*bandwidth) on the transmit spectrum mask for 802.11ax signals with the specified <bandwidth>. See 'Transmit spectrum mask OFDM, default masks' for background information.

param bandwidthD
optional repeated capability selector. Default value: Bw20 (settable in the interface 'Bw')

return
tsm_lim_yrel_lev_a: No help available

set(*tsm_lim_yrel_lev_a: float*, *bandwidthD=BandwidthD.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HEOFdm:BW
↪<bandwidth>:Y:A
driver.configure.wlanMeas.multiEval.limit.tsMask.heOfdm.bw.y.a.set(tsm_lim_yrel_
↪lev_a = 1.0, bandwidthD = repcap.BandwidthD.Default)
```

Defines the relative spectral density limit for point A (frequency offset: 2*bandwidth) on the transmit spectrum mask for 802.11ax signals with the specified <bandwidth>. See 'Transmit spectrum mask OFDM, default masks' for background information.

param tsm_lim_yrel_lev_a
No help available

param bandwidthD
optional repeated capability selector. Default value: Bw20 (settable in the interface 'Bw')

6.1.1.3.3.68 B

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HEOFdm:BW<bandwidth>:Y:B
```

class BCls

B commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthD=BandwidthD.Default*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HEOFdm:BW
↪<bandwidth>:Y:B
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.heOfdm.bw.y.b.
↪get(bandwidthD = repcap.BandwidthD.Default)
```

Defines the relative spectral density limit for point B (frequency offset: 3/2*bandwidth) on the transmit spectrum mask for 802.11ax signals with the specified <bandwidth>. See 'Transmit spectrum mask OFDM, default masks' for background information.

param bandwidthD
optional repeated capability selector. Default value: Bw20 (settable in the interface 'Bw')

return
tsm_lim_yrel_lev_b: No help available

set(*tsm_lim_yrel_lev_b: float*, *bandwidthD=BandwidthD.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HEOFdm:BW
↪<bandwidth>:Y:B
driver.configure.wlanMeas.multiEval.limit.tsMask.heOfdm.bw.y.b.set(tsm_lim_yrel_
↪lev_b = 1.0, bandwidthD = repcap.BandwidthD.Default)
```

Defines the relative spectral density limit for point B (frequency offset: 3/2*bandwidth) on the transmit spectrum mask for 802.11ax signals with the specified <bandwidth>. See 'Transmit spectrum mask OFDM, default masks' for background information.

param tsm_lim_yrel_lev_b
No help available

param bandwidthD
optional repeated capability selector. Default value: Bw20 (settable in the interface 'Bw')

6.1.1.3.3.69 C

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HEOFdm:BW<bandwidth>:Y:C
```

class CCls

C commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthD=BandwidthD.Default*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HEOFdm:BW
↪<bandwidth>:Y:C
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.heOfdm.bw.y.c.
↪get(bandwidthD = repcap.BandwidthD.Default)
```

Defines the relative spectral density limit for point C (frequency offset: 1*bandwidth) on the transmit spectrum mask for 802.11ax signals with the specified <bandwidth>. See 'Transmit spectrum mask OFDM, default masks' for background information.

param bandwidthD
optional repeated capability selector. Default value: Bw20 (settable in the interface 'Bw')

return
tsm_lim_yrel_lev_c: No help available

set(*tsm_lim_yrel_lev_c: float*, *bandwidthD=BandwidthD.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HEOFdm:BW
↪<bandwidth>:Y:C
driver.configure.wlanMeas.multiEval.limit.tsMask.heOfdm.bw.y.c.set(tsm_lim_yrel_
↪lev_c = 1.0, bandwidthD = repcap.BandwidthD.Default)
```

Defines the relative spectral density limit for point C (frequency offset: 1*bandwidth) on the transmit spectrum mask for 802.11ax signals with the specified <bandwidth>. See 'Transmit spectrum mask OFDM, default masks' for background information.

param tsm_lim_yrel_lev_c
No help available

param bandwidthD
optional repeated capability selector. Default value: Bw20 (settable in the interface 'Bw')

6.1.1.3.3.70 D

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HEOFdm:BW<bandwidth>:Y:D
```

class DCls

D commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthD=BandwidthD.Default*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HEOFdm:BW
↪<bandwidth>:Y:D
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.heOfdm.bw.y.d.
↪get(bandwidthD = repcap.BandwidthD.Default)
```

Defines the relative spectral density limit for point D (center frequency offset: 1/2*bandwidth + 1 MHz) on the transmit spectrum mask for 802.11ax signals with the specified <bandwidth>. See 'Transmit spectrum mask OFDM, default masks' for background information.

param bandwidthD
optional repeated capability selector. Default value: Bw20 (settable in the interface 'Bw')

return
tsm_lim_yrel_lev_d: No help available

set(*tsm_lim_yrel_lev_d: float*, *bandwidthD=BandwidthD.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HEOFdm:BW
↪<bandwidth>:Y:D
driver.configure.wlanMeas.multiEval.limit.tsMask.heOfdm.bw.y.d.set(tsm_lim_yrel_
↪lev_d = 1.0, bandwidthD = repcap.BandwidthD.Default)
```

Defines the relative spectral density limit for point D (center frequency offset: 1/2*bandwidth + 1 MHz) on the transmit spectrum mask for 802.11ax signals with the specified <bandwidth>. See 'Transmit spectrum mask OFDM, default masks' for background information.

param tsm_lim_yrel_lev_d
No help available

param bandwidthD
optional repeated capability selector. Default value: Bw20 (settable in the interface 'Bw')

6.1.1.3.3.71 HtOfdm

class HtOfdmCls

HtOfdm commands group definition. 6 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.tsMask.htOfdm.clone()
```

Subgroups

6.1.1.3.3.72 Bw<BandwidthC>

RepCap Settings

```
# Range: Bw5 .. Bw40
rc = driver.configure.wlanMeas.multiEval.limit.tsMask.htOfdm.bw.repcap_bandwidthC_get()
driver.configure.wlanMeas.multiEval.limit.tsMask.htOfdm.bw.repcap_bandwidthC_set(repcap.
↪BandwidthC.Bw5)
```

class BwCls

Bw commands group definition. 6 total commands, 3 Subgroups, 0 group commands Repeated Capability: BandwidthC, default value after init: BandwidthC.Bw5

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.tsMask.htOfdm.bw.clone()
```

Subgroups

6.1.1.3.3.73 AbsLimit

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HTOFdm:BW<bandwidth>
↪:ABSLimit
```

class AbsLimitCls

AbsLimit commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthC=BandwidthC.Default*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HTOFdm:BW
↪<bandwidth>:ABSLimit
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.htOfdm.bw.
↪absLimit.get(bandwidthC = repcap.BandwidthC.Default)
```

Defines an absolute power limit for 802.11n signals with the specified <bandwidth>. See 'Transmit spectrum mask OFDM, absolute limits' for background information.

param bandwidthC
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

return
tsm_lim_abs: Limit value applies to frequency offsets greater than 3/2*bandwidth, measured at 100 kHz RBW.

set(*tsm_lim_abs: float*, *bandwidthC=BandwidthC.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HTOFdm:BW
↪<bandwidth>:ABSLimit
driver.configure.wlanMeas.multiEval.limit.tsMask.htOfdm.bw.absLimit.set(tsm_lim_
↪abs = 1.0, bandwidthC = repcap.BandwidthC.Default)
```

Defines an absolute power limit for 802.11n signals with the specified <bandwidth>. See 'Transmit spectrum mask OFDM, absolute limits' for background information.

param tsm_lim_abs
Limit value applies to frequency offsets greater than 3/2*bandwidth, measured at 100 kHz RBW.

param bandwidthC
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

6.1.1.3.3.74 Band<Band>

RepCap Settings

```
# Range: Nr2 .. Nr5
rc = driver.configure.wlanMeas.multiEval.limit.tsMask.htOfdm.bw.band.repcap_band_get()
driver.configure.wlanMeas.multiEval.limit.tsMask.htOfdm.bw.band.repcap_band_set(repcap.
↪Band.Nr2)
```

class BandCls

Band commands group definition. 4 total commands, 1 Subgroups, 0 group commands Repeated Capability: Band, default value after init: Band.Nr2

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.tsMask.htOfdm.bw.band.clone()
```

Subgroups

6.1.1.3.3.75 Y

class YCls

Y commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.tsMask.htOfdm.bw.band.y.clone()
```

Subgroups

6.1.1.3.3.76 A

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HTOFdm:BW<bandwidth>:BAND
↪<band>:Y:A
```

class ACls

A commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthC=BandwidthC.Default*, *band=Band.Default*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HTOFdm:BW
↪<bandwidth>:BAND<band>:Y:A
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.htOfdm.bw.band.
↪y.a.get(bandwidthC = repcap.BandwidthC.Default, band = repcap.Band.Default)
```

Defines the relative spectral density limit for point A (frequency offset: 2*bandwidth) on the transmit spectrum mask for 802.11n signals with the specified <bandwidth> and the selected <band>. See 'Transmit spectrum mask OFDM, default masks' for background information.

param bandwidthC
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

param band
optional repeated capability selector. Default value: Nr2 (settable in the interface 'Band')

return
tsm_lim_yrel_lev_a: No help available

set(*tsm_lim_yrel_lev_a: float*, *bandwidthC=BandwidthC.Default*, *band=Band.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HTOFdm:BW
↪<bandwidth>:BAND<band>:Y:A
driver.configure.wlanMeas.multiEval.limit.tsMask.htOfdm.bw.band.y.a.set(tsm_lim_
↪yrel_lev_a = 1.0, bandwidthC = repcap.BandwidthC.Default, band = repcap.Band.
↪Default)
```

Defines the relative spectral density limit for point A (frequency offset: 2*bandwidth) on the transmit spectrum mask for 802.11n signals with the specified <bandwidth> and the selected <band>. See 'Transmit spectrum mask OFDM, default masks' for background information.

param tsm_lim_yrel_lev_a
No help available

param bandwidthC
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

param band
optional repeated capability selector. Default value: Nr2 (settable in the interface 'Band')

6.1.1.3.3.77 B

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HTOFdm:BW<bandwidth>:BAND
↪<band>:Y:B
```

class BCls

B commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthC=BandwidthC.Default*, *band=Band.Default*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HTOFdm:BW
↪<bandwidth>:BAND<band>:Y:B
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.htOfdm.bw.band.
↪y.b.get(bandwidthC = repcap.BandwidthC.Default, band = repcap.Band.Default)
```

Defines the relative spectral density limit for point B (frequency offset: 3/2*bandwidth) on the transmit spectrum mask for 802.11n signals with the specified <bandwidth> and the selected <band>. See 'Transmit spectrum mask OFDM, default masks' for background information.

param bandwidthC
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

param band
optional repeated capability selector. Default value: Nr2 (settable in the interface 'Band')

return
tsm_lim_yrel_lev_b: No help available

set(*tsm_lim_yrel_lev_b: float*, *bandwidthC=BandwidthC.Default*, *band=Band.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HTOFdm:BW
↪<bandwidth>:BAND<band>:Y:B
driver.configure.wlanMeas.multiEval.limit.tsMask.htOfdm.bw.band.y.b.set(tsm_lim_
↪yrel_lev_b = 1.0, bandwidthC = repcap.BandwidthC.Default, band = repcap.Band.
↪Default)
```

Defines the relative spectral density limit for point B (frequency offset: 3/2*bandwidth) on the transmit spectrum mask for 802.11n signals with the specified <bandwidth> and the selected <band>. See 'Transmit spectrum mask OFDM, default masks' for background information.

param tsm_lim_yrel_lev_b
No help available

param bandwidthC
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

param band
optional repeated capability selector. Default value: Nr2 (settable in the interface 'Band')

6.1.1.3.3.78 C

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HTOFdm:BW<bandwidth>:BAND
↪<band>:Y:C
```

class CCls

C commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthC=BandwidthC.Default*, *band=Band.Default*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HTOFdm:BW
↪<bandwidth>:BAND<band>:Y:C
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.htOfdm.bw.band.
↪y.c.get(bandwidthC = repcap.BandwidthC.Default, band = repcap.Band.Default)
```

Defines the relative spectral density limit for point C (frequency offset: 1*bandwidth) on the transmit spectrum mask for 802.11n signals with the specified <bandwidth> and the selected <band>. See 'Transmit spectrum mask OFDM, default masks' for background information.

param bandwidthC
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

param band
optional repeated capability selector. Default value: Nr2 (settable in the interface 'Band')

return
tsm_lim_yrel_lev_c: No help available

set(*tsm_lim_yrel_lev_c: float*, *bandwidthC=BandwidthC.Default*, *band=Band.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HTOFdm:BW
↪<bandwidth>:BAND<band>:Y:C
driver.configure.wlanMeas.multiEval.limit.tsMask.htOfdm.bw.band.y.c.set(tsm_lim_
↪yrel_lev_c = 1.0, bandwidthC = repcap.BandwidthC.Default, band = repcap.Band.
↪Default)
```

Defines the relative spectral density limit for point C (frequency offset: 1*bandwidth) on the transmit spectrum mask for 802.11n signals with the specified <bandwidth> and the selected <band>. See 'Transmit spectrum mask OFDM, default masks' for background information.

param tsm_lim_yrel_lev_c
No help available

param bandwidthC
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

param band
optional repeated capability selector. Default value: Nr2 (settable in the interface 'Band')

6.1.1.3.3.79 D

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HTOFdm:BW<bandwidth>:BAND
↪<band>:Y:D
```

class DCls

D commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthC=BandwidthC.Default*, *band=Band.Default*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HTOFdm:BW
↪<bandwidth>:BAND<band>:Y:D
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.htOfdm.bw.band.
↪y.d.get(bandwidthC = repcap.BandwidthC.Default, band = repcap.Band.Default)
```

Defines the relative spectral density limit for point D (frequency offset: 1/2*bandwidth + 1 MHz) on the transmit spectrum mask for 802.11n signals with the specified <bandwidth> and the selected <band>. See 'Transmit spectrum mask OFDM, default masks' for background information.

param bandwidthC
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

param band
optional repeated capability selector. Default value: Nr2 (settable in the interface 'Band')

return
tsm_lim_yrel_lev_d: No help available

set(*tsm_lim_yrel_lev_d: float*, *bandwidthC=BandwidthC.Default*, *band=Band.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HTOFdm:BW
↪<bandwidth>:BAND<band>:Y:D
driver.configure.wlanMeas.multiEval.limit.tsMask.htOfdm.bw.band.y.d.set(tsm_lim_
↪yrel_lev_d = 1.0, bandwidthC = repcap.BandwidthC.Default, band = repcap.Band.
↪Default)
```

Defines the relative spectral density limit for point D (frequency offset: 1/2*bandwidth + 1 MHz) on the transmit spectrum mask for 802.11n signals with the specified <bandwidth> and the selected <band>. See 'Transmit spectrum mask OFDM, default masks' for background information.

param tsm_lim_yrel_lev_d
No help available

param bandwidthC
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

param band
optional repeated capability selector. Default value: Nr2 (settable in the interface 'Band')

6.1.1.3.3.80 Enable

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HTOFdm:BW<bandwidth>:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthC=BandwidthC.Default*) → bool

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HTOFdm:BW
↪<bandwidth>:ENABle
value: bool = driver.configure.wlanMeas.multiEval.limit.tsMask.htOfdm.bw.enable.
↪get(bandwidthC = repcap.BandwidthC.Default)
```

Enables or disables the transmit spectrum mask for 802.11n signals with the specified <bandwidth>, i.e. activates or deactivates the corresponding limit check.

param bandwidthC
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

return
tsm_lim_enable: No help available

set(*tsm_lim_enable: bool*, *bandwidthC=BandwidthC.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HTOFdm:BW
↪<bandwidth>:ENABle
driver.configure.wlanMeas.multiEval.limit.tsMask.htOfdm.bw.enable.set(tsm_lim_
↪enable = False, bandwidthC = repcap.BandwidthC.Default)
```

Enables or disables the transmit spectrum mask for 802.11n signals with the specified <bandwidth>, i.e. activates or deactivates the corresponding limit check.

param tsm_lim_enable
No help available

param bandwidthC
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

6.1.1.3.3.81 Lofdm

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:LOFDm:ENABle
```

class LofdmCls

Lofdm commands group definition. 5 total commands, 1 Subgroups, 1 group commands

get_enable() → bool

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:TSMask:LOFDm:ENABle
value: bool = driver.configure.wlanMeas.multiEval.limit.tsMask.lofdm.get_
↪enable()
```

No command help available

return
tsm_lim_enable: No help available

set_enable(*tsm_lim_enable: bool*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>
↪:MEValuation:LIMit:TSMask:LOFDm:ENABle
driver.configure.wlanMeas.multiEval.limit.tsMask.lofdm.set_enable(tsm_lim_
↪enable = False)
```

No command help available

param tsm_lim_enable
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.tsMask.lofdm.clone()
```

Subgroups

6.1.1.3.3.82 Y

SCPI Commands :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:LOFDm:Y:A
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:LOFDm:Y:B
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:LOFDm:Y:C
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:LOFDm:Y:D
```

class YCls

Y commands group definition. 4 total commands, 0 Subgroups, 4 group commands

get_a() → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:LOFDm:Y:A
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.lofdm.y.get_a()
```

No command help available

return
tsm_lim_yrel_lev_a: No help available

get_b() → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:LOFDm:Y:B
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.lofdm.y.get_b()
```

No command help available

return
tsm_lim_yrel_lev_b: No help available

get_c() → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:LOFDm:Y:C
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.lofdm.y.get_c()
```

No command help available

return
tsm_lim_yrel_lev_c: No help available

get_d() → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:LOFDm:Y:D
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.lofdm.y.get_d()
```

No command help available

return
tsm_lim_yrel_lev_d: No help available

set_a(*tsm_lim_yrel_lev_a: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:LOFDm:Y:A
driver.configure.wlanMeas.multiEval.limit.tsMask.lofdm.y.set_a(tsm_lim_yrel_lev_
↪a = 1.0)
```

No command help available

param tsm_lim_yrel_lev_a
No help available

set_b(*tsm_lim_yrel_lev_b: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:LOFDm:Y:B
driver.configure.wlanMeas.multiEval.limit.tsMask.lofdm.y.set_b(tsm_lim_yrel_lev_
↪b = 1.0)
```

No command help available

param tsm_lim_yrel_lev_b
No help available

set_c(*tsm_lim_yrel_lev_c: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:LOFDm:Y:C
driver.configure.wlanMeas.multiEval.limit.tsMask.lofdm.y.set_c(tsm_lim_yrel_lev_
↪c = 1.0)
```

No command help available

param tsm_lim_yrel_lev_c
No help available

set_d(*tsm_lim_yrel_lev_d: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:LOFDm:Y:D
driver.configure.wlanMeas.multiEval.limit.tsMask.lofdm.y.set_d(tsm_lim_yrel_lev_
↪d = 1.0)
```

No command help available

param tsm_lim_yrel_lev_d
No help available

6.1.1.3.3.83 Pofdm

class PofdmCls

Pofdm commands group definition. 22 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.clone()
```

Subgroups

6.1.1.3.3.84 Bw

class BwCls

Bw commands group definition. 22 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.clone()
```

Subgroups

6.1.1.3.3.85 Absolute

class AbsoluteCls

Absolute commands group definition. 6 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.absolute.clone()
```

Subgroups

6.1.1.3.3.86 Y

class YCls

Y commands group definition. 6 total commands, 6 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.absolute.y.clone()
```

Subgroups

6.1.1.3.3.87 A

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW<bandwidth>
↪:ABSolute:Y:A
```

class ACls

A commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthA=BandwidthA.Bw10*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:ABSolute:Y:A
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.
↪absolute.y.a.get(bandwidthA = repcap.BandwidthA.Bw10)
```

Defines the Y-value of point A (f = 2 <BW>) on the ETSI ITS absolute emission mask for the specified <bandwidth>. For background information, see 'Transmit spectrum mask OFDM, absolute limits'.

param bandwidthA
optional repeated capability selector. Default value: Bw10

return
tsm_lim_yabs_lev_a: No help available

set(*tsm_lim_yabs_lev_a: float*, *bandwidthA=BandwidthA.Bw10*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:ABSolute:Y:A
driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.absolute.y.a.set(tsm_
↪lim_yabs_lev_a = 1.0, bandwidthA = repcap.BandwidthA.Bw10)
```

Defines the Y-value of point A (f = 2 <BW>) on the ETSI ITS absolute emission mask for the specified <bandwidth>. For background information, see 'Transmit spectrum mask OFDM, absolute limits'.

param tsm_lim_yabs_lev_a
No help available

param bandwidthA
optional repeated capability selector. Default value: Bw10

6.1.1.3.3.88 B

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW<bandwidth>
↪:ABSolute:Y:B
```

class BCls

B commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthA=BandwidthA.Bw10*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:ABSolute:Y:B
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.
↪absolute.y.b.get(bandwidthA = repcap.BandwidthA.Bw10)
```

Sets/queries the Y-value of point B (f = 1.5 <BW>) on the ETSI ITS absolute emission mask for the specified <bandwidth>. For background information, see 'Transmit spectrum mask OFDM, absolute limits'.

param bandwidthA
optional repeated capability selector. Default value: Bw10

return
tsm_lim_yabs_lev_b: No help available

set(*tsm_lim_yabs_lev_b: float*, *bandwidthA=BandwidthA.Bw10*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:ABSolute:Y:B
driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.absolute.y.b.set(tsm_
↪lim_yabs_lev_b = 1.0, bandwidthA = repcap.BandwidthA.Bw10)
```

Sets/queries the Y-value of point B (f = 1.5 <BW>) on the ETSI ITS absolute emission mask for the specified <bandwidth>. For background information, see 'Transmit spectrum mask OFDM, absolute limits'.

param tsm_lim_yabs_lev_b
No help available

param bandwidthA
optional repeated capability selector. Default value: Bw10

6.1.1.3.3.89 C

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW<bandwidth>
↪:ABSolute:Y:C
```

class CCls

C commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthA=BandwidthA.Bw10*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:ABSolute:Y:C
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.
↪absolute.y.c.get(bandwidthA = repcap.BandwidthA.Bw10)
```

Defines the Y-value of point C (f = <BW>) on the ETSI ITS absolute emission mask for the specified <bandwidth>. For background information, see 'Transmit spectrum mask OFDM, absolute limits'.

param bandwidthA
optional repeated capability selector. Default value: Bw10

return
tsm_lim_yabs_lev_c: No help available

set(*tsm_lim_yabs_lev_c: float*, *bandwidthA=BandwidthA.Bw10*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:ABSolute:Y:C
driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.absolute.y.c.set(tsm_
↪lim_yabs_lev_c = 1.0, bandwidthA = repcap.BandwidthA.Bw10)
```

Defines the Y-value of point C (f = <BW>) on the ETSI ITS absolute emission mask for the specified <bandwidth>. For background information, see 'Transmit spectrum mask OFDM, absolute limits'.

param tsm_lim_yabs_lev_c
No help available

param bandwidthA
optional repeated capability selector. Default value: Bw10

6.1.1.3.3.90 D

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW<bandwidth>
↪:ABSolute:Y:D
```

class DCls

D commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthA=BandwidthA.Bw10*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:ABSolute:Y:D
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.
↪absolute.y.d.get(bandwidthA = repcap.BandwidthA.Bw10)
```

Defines the Y-value of point D (f = 0.55 <BW>) on the ETSI ITS absolute emission mask for the specified <bandwidth>. For background information, see 'Transmit spectrum mask OFDM, absolute limits'.

param bandwidthA
optional repeated capability selector. Default value: Bw10

return
tsm_lim_yabs_lev_d: No help available

set(*tsm_lim_yabs_lev_d: float*, *bandwidthA=BandwidthA.Bw10*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:ABSolute:Y:D
driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.absolute.y.d.set(tsm_
↪lim_yabs_lev_d = 1.0, bandwidthA = repcap.BandwidthA.Bw10)
```

Defines the Y-value of point D (f = 0.55 <BW>) on the ETSI ITS absolute emission mask for the specified <bandwidth>. For background information, see 'Transmit spectrum mask OFDM, absolute limits'.

param tsm_lim_yabs_lev_d
No help available

param bandwidthA
optional repeated capability selector. Default value: Bw10

6.1.1.3.3.91 E

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW<bandwidth>
↪:ABSolute:Y:E
```

class ECls

E commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthA=BandwidthA.Bw10*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:ABSolute:Y:E
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.
↪absolute.y.e.get(bandwidthA = repcap.BandwidthA.Bw10)
```

Defines the Y-value of point E (f = 0.5 <BW>) on the ETSI ITS absolute emission mask for the specified <bandwidth>. For background information, see 'Transmit spectrum mask OFDM, absolute limits'.

param bandwidthA
optional repeated capability selector. Default value: Bw10

return
tsm_lim_yabs_lev_e: No help available

set(*tsm_lim_yabs_lev_e: float*, *bandwidthA=BandwidthA.Bw10*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:ABSolute:Y:E
driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.absolute.y.e.set(tsm_
↪lim_yabs_lev_e = 1.0, bandwidthA = repcap.BandwidthA.Bw10)
```

Defines the Y-value of point E (f = 0.5 <BW>) on the ETSI ITS absolute emission mask for the specified <bandwidth>. For background information, see 'Transmit spectrum mask OFDM, absolute limits'.

param tsm_lim_yabs_lev_e
No help available

param bandwidthA
optional repeated capability selector. Default value: Bw10

6.1.1.3.3.92 F

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW<bandwidth>
↪:ABSolute:Y:F
```

class FCls

F commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthA=BandwidthA.Bw10*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:ABSolute:Y:F
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.
↪absolute.y.f.get(bandwidthA = repcap.BandwidthA.Bw10)
```

Defines the Y-value of point F (f = 0.45 <BW>) on the ETSI ITS absolute emission mask for the specified <bandwidth>. For background information, see 'Transmit spectrum mask OFDM, absolute limits'.

param bandwidthA
optional repeated capability selector. Default value: Bw10

return
tsm_lim_yabs_lev_f: No help available

set(*tsm_lim_yabs_lev_f: float*, *bandwidthA=BandwidthA.Bw10*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:ABSolute:Y:F
driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.absolute.y.f.set(tsm_
↪lim_yabs_lev_f = 1.0, bandwidthA = repcap.BandwidthA.Bw10)
```

Defines the Y-value of point F (f = 0.45 <BW>) on the ETSI ITS absolute emission mask for the specified <bandwidth>. For background information, see 'Transmit spectrum mask OFDM, absolute limits'.

param tsm_lim_yabs_lev_f
No help available

param bandwidthA
optional repeated capability selector. Default value: Bw10

6.1.1.3.3.93 Ca

class CaCls

Ca commands group definition. 5 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.ca.clone()
```

Subgroups

6.1.1.3.3.94 Y

class YCls

Y commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.ca.y.clone()
```

Subgroups

6.1.1.3.3.95 A

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW<bandwidth>:CA:Y:A
```

class ACls

A commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthB=BandwidthB.Bw5*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:CA:Y:A
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.ca.y.a.
↪get(bandwidthB = repcap.BandwidthB.Bw5)
```

Defines the Y-value of point A (f = 2 <bandwidth>) on the 802.11p spectrum mask for power class A and the specified <bandwidth>. For background information, see 'Transmit spectrum mask OFDM, by regulation'.

param bandwidthB
optional repeated capability selector. Default value: Bw5

return
tsm_lim_yrel_lev_a: No help available

set(*tsm_lim_yrel_lev_a: float*, *bandwidthB=BandwidthB.Bw5*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:CA:Y:A
driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.ca.y.a.set(tsm_lim_
↪yrel_lev_a = 1.0, bandwidthB = repcap.BandwidthB.Bw5)
```

Defines the Y-value of point A (f = 2 <bandwidth>) on the 802.11p spectrum mask for power class A and the specified <bandwidth>. For background information, see 'Transmit spectrum mask OFDM, by regulation'.

param tsm_lim_yrel_lev_a
No help available

param bandwidthB
optional repeated capability selector. Default value: Bw5

6.1.1.3.3.96 B

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW<bandwidth>:CA:Y:B
```

class BCls

B commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthB=BandwidthB.Bw5*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:CA:Y:B
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.ca.y.b.
↪get(bandwidthB = repcap.BandwidthB.Bw5)
```

Defines the Y-value of point B (f = 1.5 <bandwidth>) on the 802.11p spectrum mask for power class A and the specified <bandwidth>. For background information, see 'Transmit spectrum mask OFDM, by regulation'.

param bandwidthB
optional repeated capability selector. Default value: Bw5

return
tsm_lim_yrel_lev_b: No help available

set(*tsm_lim_yrel_lev_b: float*, *bandwidthB=BandwidthB.Bw5*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:CA:Y:B
driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.ca.y.b.set(tsm_lim_
↪yrel_lev_b = 1.0, bandwidthB = repcap.BandwidthB.Bw5)
```

Defines the Y-value of point B (f = 1.5 <bandwidth>) on the 802.11p spectrum mask for power class A and the specified <bandwidth>. For background information, see 'Transmit spectrum mask OFDM, by regulation'.

param tsm_lim_yrel_lev_b
No help available

param bandwidthB
optional repeated capability selector. Default value: Bw5

6.1.1.3.3.97 C

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW<bandwidth>:CA:Y:C
```

class CCls

C commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthB=BandwidthB.Bw5*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:CA:Y:C
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.ca.y.c.
↪get(bandwidthB = repcap.BandwidthB.Bw5)
```

Defines the Y-value of point C (f = <bandwidth>) on the 802.11p spectrum mask for power class A and the specified <bandwidth>. For background information, see 'Transmit spectrum mask OFDM, by regulation'.

param bandwidthB
optional repeated capability selector. Default value: Bw5

return
tsm_lim_yrel_lev_c: No help available

set(*tsm_lim_yrel_lev_c: float*, *bandwidthB=BandwidthB.Bw5*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:CA:Y:C
driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.ca.y.c.set(tsm_lim_
↪yrel_lev_c = 1.0, bandwidthB = repcap.BandwidthB.Bw5)
```

Defines the Y-value of point C (f = <bandwidth>) on the 802.11p spectrum mask for power class A and the specified <bandwidth>. For background information, see 'Transmit spectrum mask OFDM, by regulation'.

param tsm_lim_yrel_lev_c
No help available

param bandwidthB
optional repeated capability selector. Default value: Bw5

6.1.1.3.3.98 D

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW<bandwidth>:CA:Y:D
```

class DCls

D commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthB=BandwidthB.Bw5*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:CA:Y:D
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.ca.y.d.
↪get(bandwidthB = repcap.BandwidthB.Bw5)
```

Defines the Y-value of point D (f = 0.55 <bandwidth>) on the 802.11p spectrum mask for power class A and the specified <bandwidth>. For background information, see 'Transmit spectrum mask OFDM, by regulation'.

param bandwidthB
optional repeated capability selector. Default value: Bw5

return
tsm_lim_yrel_lev_d: No help available

set(*tsm_lim_yrel_lev_d: float*, *bandwidthB=BandwidthB.Bw5*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:CA:Y:D
driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.ca.y.d.set(tsm_lim_
↪yrel_lev_d = 1.0, bandwidthB = repcap.BandwidthB.Bw5)
```

Defines the Y-value of point D (f = 0.55 <bandwidth>) on the 802.11p spectrum mask for power class A and the specified <bandwidth>. For background information, see 'Transmit spectrum mask OFDM, by regulation'.

param tsm_lim_yrel_lev_d
No help available

param bandwidthB
optional repeated capability selector. Default value: Bw5

6.1.1.3.3.99 E

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW<bandwidth>:CA:Y:E
```

class ECls

E commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthB=BandwidthB.Bw5*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:CA:Y:E
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.ca.y.e.
↪get(bandwidthB = repcap.BandwidthB.Bw5)
```

Defines the Y-value of point E (f = 0.5 <bandwidth>) on the 802.11p spectrum mask for power class A and the specified <bandwidth>. For background information, see 'Transmit spectrum mask OFDM, by regulation'.

param bandwidthB
optional repeated capability selector. Default value: Bw5

return
tsm_lim_yrel_lev_e: No help available

set(*tsm_lim_yrel_lev_e: float*, *bandwidthB=BandwidthB.Bw5*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:CA:Y:E
driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.ca.y.e.set(tsm_lim_
↪yrel_lev_e = 1.0, bandwidthB = repcap.BandwidthB.Bw5)
```

Defines the Y-value of point E (f = 0.5 <bandwidth>) on the 802.11p spectrum mask for power class A and the specified <bandwidth>. For background information, see 'Transmit spectrum mask OFDM, by regulation'.

param tsm_lim_yrel_lev_e
No help available

param bandwidthB
optional repeated capability selector. Default value: Bw5

6.1.1.3.3.100 Cb

class CbCls

Cb commands group definition. 5 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.cb.clone()
```

Subgroups

6.1.1.3.3.101 Y

class YCls

Y commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.cb.y.clone()
```

Subgroups

6.1.1.3.3.102 A

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW<bandwidth>:CB:Y:A
```

class ACls

A commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthB=BandwidthB.Bw5*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:CB:Y:A
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.cb.y.a.
↪get(bandwidthB = repcap.BandwidthB.Bw5)
```

Defines the Y-value of point A (f = 2 <bandwidth>) on the 802.11p spectrum mask for power class B and the specified <bandwidth>. For background information, see 'Transmit spectrum mask OFDM, by regulation'.

param bandwidthB
optional repeated capability selector. Default value: Bw5

return
tsm_lim_yrel_lev_a: No help available

set(*tsm_lim_yrel_lev_a: float*, *bandwidthB=BandwidthB.Bw5*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:CB:Y:A
driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.cb.y.a.set(tsm_lim_
↪yrel_lev_a = 1.0, bandwidthB = repcap.BandwidthB.Bw5)
```

Defines the Y-value of point A (f = 2 <bandwidth>) on the 802.11p spectrum mask for power class B and the specified <bandwidth>. For background information, see 'Transmit spectrum mask OFDM, by regulation'.

param tsm_lim_yrel_lev_a
No help available

param bandwidthB
optional repeated capability selector. Default value: Bw5

6.1.1.3.3.103 B

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW<bandwidth>:CB:Y:B
```

class BCls

B commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthB=BandwidthB.Bw5*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:CB:Y:B
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.cb.y.b.
↪get(bandwidthB = repcap.BandwidthB.Bw5)
```

Defines the Y-value of point B (f = 1.5 <bandwidth>) on the 802.11p spectrum mask for power class B and the specified <bandwidth>. For background information, see 'Transmit spectrum mask OFDM, by regulation'.

param bandwidthB
optional repeated capability selector. Default value: Bw5

return
tsm_lim_yrel_lev_b: No help available

set(*tsm_lim_yrel_lev_b: float*, *bandwidthB=BandwidthB.Bw5*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:CB:Y:B
driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.cb.y.b.set(tsm_lim_
↪yrel_lev_b = 1.0, bandwidthB = repcap.BandwidthB.Bw5)
```

Defines the Y-value of point B (f = 1.5 <bandwidth>) on the 802.11p spectrum mask for power class B and the specified <bandwidth>. For background information, see 'Transmit spectrum mask OFDM, by regulation'.

param tsm_lim_yrel_lev_b
No help available

param bandwidthB
optional repeated capability selector. Default value: Bw5

6.1.1.3.3.104 C

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW<bandwidth>:CB:Y:C
```

class CCls

C commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthB=BandwidthB.Bw5*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:CB:Y:C
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.cb.y.c.
↪get(bandwidthB = repcap.BandwidthB.Bw5)
```

Defines the Y-value of point C (f = <bandwidth>) on the 802.11p spectrum mask for power class B and the specified <bandwidth>. For background information, see 'Transmit spectrum mask OFDM, by regulation'.

param bandwidthB
optional repeated capability selector. Default value: Bw5

return
tsm_lim_yrel_lev_c: No help available

set(*tsm_lim_yrel_lev_c: float*, *bandwidthB=BandwidthB.Bw5*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:CB:Y:C
driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.cb.y.c.set(tsm_lim_
↪yrel_lev_c = 1.0, bandwidthB = repcap.BandwidthB.Bw5)
```

Defines the Y-value of point C (f = <bandwidth>) on the 802.11p spectrum mask for power class B and the specified <bandwidth>. For background information, see 'Transmit spectrum mask OFDM, by regulation'.

param tsm_lim_yrel_lev_c
No help available

param bandwidthB
optional repeated capability selector. Default value: Bw5

6.1.1.3.3.105 D

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW<bandwidth>:CB:Y:D
```

class DCls

D commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthB=BandwidthB.Bw5*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:CB:Y:D
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.cb.y.d.
↪get(bandwidthB = repcap.BandwidthB.Bw5)
```

Defines the Y-value of point D (f = 0.55 <bandwidth>) on the 802.11p spectrum mask for power class B and the specified <bandwidth>. For background information, see 'Transmit spectrum mask OFDM, by regulation'.

param bandwidthB
optional repeated capability selector. Default value: Bw5

return
tsm_lim_yrel_lev_d: No help available

set(*tsm_lim_yrel_lev_d: float*, *bandwidthB=BandwidthB.Bw5*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:CB:Y:D
driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.cb.y.d.set(tsm_lim_
↪yrel_lev_d = 1.0, bandwidthB = repcap.BandwidthB.Bw5)
```

Defines the Y-value of point D (f = 0.55 <bandwidth>) on the 802.11p spectrum mask for power class B and the specified <bandwidth>. For background information, see 'Transmit spectrum mask OFDM, by regulation'.

param tsm_lim_yrel_lev_d
No help available

param bandwidthB
optional repeated capability selector. Default value: Bw5

6.1.1.3.3.106 E

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW<bandwidth>:CB:Y:E
```

class ECls

E commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthB=BandwidthB.Bw5*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:CB:Y:E
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.cb.y.e.
↪get(bandwidthB = repcap.BandwidthB.Bw5)
```

Defines the Y-value of point E (f = 0.5 <bandwidth>) on the 802.11p spectrum mask for power class B and the specified <bandwidth>. For background information, see 'Transmit spectrum mask OFDM, by regulation'.

param bandwidthB
optional repeated capability selector. Default value: Bw5

return
tsm_lim_yrel_lev_e: No help available

set(*tsm_lim_yrel_lev_e: float*, *bandwidthB=BandwidthB.Bw5*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:CB:Y:E
driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.cb.y.e.set(tsm_lim_
↪yrel_lev_e = 1.0, bandwidthB = repcap.BandwidthB.Bw5)
```

Defines the Y-value of point E (f = 0.5 <bandwidth>) on the 802.11p spectrum mask for power class B and the specified <bandwidth>. For background information, see 'Transmit spectrum mask OFDM, by regulation'.

param tsm_lim_yrel_lev_e
No help available

param bandwidthB
optional repeated capability selector. Default value: Bw5

6.1.1.3.3.107 Enable

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW<bandwidth>:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthB=BandwidthB.Bw5*) → bool

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:ENABle
value: bool = driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.enable.
↪get(bandwidthB = repcap.BandwidthB.Bw5)
```

Activates or deactivates the transmit spectrum mask limit check for 802.11p signals with the specified <bandwidth>.

param bandwidthB
optional repeated capability selector. Default value: Bw5

return
tsm_lim_enable: No help available

set(*tsm_lim_enable: bool*, *bandwidthB=BandwidthB.Bw5*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:ENABle
driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.enable.set(tsm_lim_
↪enable = False, bandwidthB = repcap.BandwidthB.Bw5)
```

Activates or deactivates the transmit spectrum mask limit check for 802.11p signals with the specified <bandwidth>.

param tsm_lim_enable
No help available

param bandwidthB
optional repeated capability selector. Default value: Bw5

6.1.1.3.3.108 UserDefined

class UserDefinedCls

UserDefined commands group definition. 5 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.userDefined.clone()
```

Subgroups

6.1.1.3.3.109 Y

class YCls

Y commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.userDefined.y.clone()
```

Subgroups

6.1.1.3.3.110 A

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW<bandwidth>
↪:UDEFined:Y:A
```

class ACls

A commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthB=BandwidthB.Bw5*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:UDEFined:Y:A
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.
↪userDefined.y.a.get(bandwidthB = repcap.BandwidthB.Bw5)
```

Defines the Y-value of point A (f = 2 <bandwidth>) on the 802.11p spectrum mask for a user-defined power class and the specified <bandwidth>. For background information, see 'Transmit spectrum mask OFDM, by regulation'.

param bandwidthB
optional repeated capability selector. Default value: Bw5

return
tsm_lim_yrel_lev_a: No help available

set(*tsm_lim_yrel_lev_a: float*, *bandwidthB=BandwidthB.Bw5*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:UDEFined:Y:A
driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.userDefined.y.a.
↪set(tsm_lim_yrel_lev_a = 1.0, bandwidthB = repcap.BandwidthB.Bw5)
```

Defines the Y-value of point A (f = 2 <bandwidth>) on the 802.11p spectrum mask for a user-defined power class and the specified <bandwidth>. For background information, see 'Transmit spectrum mask OFDM, by regulation'.

param tsm_lim_yrel_lev_a
No help available

param bandwidthB
optional repeated capability selector. Default value: Bw5

6.1.1.3.3.111 B

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW<bandwidth>
↪:UDEFined:Y:B
```

class BCls

B commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthB=BandwidthB.Bw5*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:UDEFined:Y:B
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.
↪userDefined.y.b.get(bandwidthB = repcap.BandwidthB.Bw5)
```

Defines the Y-value of point B (f = 1.5 <bandwidth>) on the 802.11p spectrum mask for a user-defined power class and the specified <bandwidth>. For background information, see 'Transmit spectrum mask OFDM, by regulation'.

param bandwidthB
optional repeated capability selector. Default value: Bw5

return
tsm_lim_yrel_lev_b: No help available

set(*tsm_lim_yrel_lev_b: float*, *bandwidthB=BandwidthB.Bw5*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:UDEFined:Y:B
driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.userDefined.y.b.
↪set(tsm_lim_yrel_lev_b = 1.0, bandwidthB = repcap.BandwidthB.Bw5)
```

Defines the Y-value of point B (f = 1.5 <bandwidth>) on the 802.11p spectrum mask for a user-defined power class and the specified <bandwidth>. For background information, see 'Transmit spectrum mask OFDM, by regulation'.

param tsm_lim_yrel_lev_b
No help available

param bandwidthB
optional repeated capability selector. Default value: Bw5

6.1.1.3.3.112 C

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW<bandwidth>
↪:UDEFined:Y:C
```

class CCls

C commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthB=BandwidthB.Bw5*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:UDEFined:Y:C
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.
↪userDefined.y.c.get(bandwidthB = repcap.BandwidthB.Bw5)
```

Defines the Y-value of point C (f = <bandwidth>) on the 802.11p spectrum mask for a user-defined power class and the specified <bandwidth>. For background information, see 'Transmit spectrum mask OFDM, by regulation'.

param bandwidthB
optional repeated capability selector. Default value: Bw5

return
tsm_lim_yrel_lev_c: No help available

set(*tsm_lim_yrel_lev_c: float*, *bandwidthB=BandwidthB.Bw5*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:UDEFined:Y:C
driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.userDefined.y.c.
↪set(tsm_lim_yrel_lev_c = 1.0, bandwidthB = repcap.BandwidthB.Bw5)
```

Defines the Y-value of point C (f = <bandwidth>) on the 802.11p spectrum mask for a user-defined power class and the specified <bandwidth>. For background information, see 'Transmit spectrum mask OFDM, by regulation'.

param tsm_lim_yrel_lev_c
No help available

param bandwidthB
optional repeated capability selector. Default value: Bw5

6.1.1.3.3.113 D

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW<bandwidth>
↪:UDEFined:Y:D
```

class DCls

D commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthB=BandwidthB.Bw5*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:UDEFined:Y:D
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.
↪userDefined.y.d.get(bandwidthB = repcap.BandwidthB.Bw5)
```

Defines the Y-value of point D (f = 0.55 <bandwidth>) on the 802.11p spectrum mask for a user-defined power class and the specified <bandwidth>. For background information, see 'Transmit spectrum mask OFDM, by regulation'.

param bandwidthB
optional repeated capability selector. Default value: Bw5

return
tsm_lim_yrel_lev_d: No help available

set(*tsm_lim_yrel_lev_d: float*, *bandwidthB=BandwidthB.Bw5*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:UDEFined:Y:D
driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.userDefined.y.d.
↪set(tsm_lim_yrel_lev_d = 1.0, bandwidthB = repcap.BandwidthB.Bw5)
```

Defines the Y-value of point D (f = 0.55 <bandwidth>) on the 802.11p spectrum mask for a user-defined power class and the specified <bandwidth>. For background information, see 'Transmit spectrum mask OFDM, by regulation'.

param tsm_lim_yrel_lev_d
No help available

param bandwidthB
optional repeated capability selector. Default value: Bw5

6.1.1.3.3.114 E

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW<bandwidth>
↪:UDEFined:Y:E
```

class ECls

E commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthB=BandwidthB.Bw5*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:UDEFined:Y:E
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.
↪userDefined.y.e.get(bandwidthB = repcap.BandwidthB.Bw5)
```

Defines the Y-value of point E (f = 0.5 <bandwidth>) on the 802.11p spectrum mask for a user-defined power class and the specified <bandwidth>. For background information, see 'Transmit spectrum mask OFDM, by regulation'.

param bandwidthB
optional repeated capability selector. Default value: Bw5

return
tsm_lim_yrel_lev_e: No help available

set(*tsm_lim_yrel_lev_e: float*, *bandwidthB=BandwidthB.Bw5*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW
↪<bandwidth>:UDEFined:Y:E
driver.configure.wlanMeas.multiEval.limit.tsMask.pofdm.bw.userDefined.y.e.
↪set(tsm_lim_yrel_lev_e = 1.0, bandwidthB = repcap.BandwidthB.Bw5)
```

Defines the Y-value of point E (f = 0.5 <bandwidth>) on the 802.11p spectrum mask for a user-defined power class and the specified <bandwidth>. For background information, see 'Transmit spectrum mask OFDM, by regulation'.

param tsm_lim_yrel_lev_e
No help available

param bandwidthB
optional repeated capability selector. Default value: Bw5

6.1.1.3.3.115 VhtOfdm

class VhtOfdmCls

VhtOfdm commands group definition. 6 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.tsMask.vhtOfdm.clone()
```

Subgroups

6.1.1.3.3.116 Bw<BandwidthE>

RepCap Settings

```
# Range: Bw5 .. Bw8080
rc = driver.configure.wlanMeas.multiEval.limit.tsMask.vht0fdm.bw.repcap_bandwidthE_get()
driver.configure.wlanMeas.multiEval.limit.tsMask.vht0fdm.bw.repcap_bandwidthE_set(repcap.
↪BandwidthE.Bw5)
```

class BwCls

Bw commands group definition. 6 total commands, 3 Subgroups, 0 group commands Repeated Capability: BandwidthE, default value after init: BandwidthE.Bw5

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.tsMask.vht0fdm.bw.clone()
```

Subgroups

6.1.1.3.3.117 AbsLimit

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:VHTofdm:BW<bandwidth>
↪:ABSLimit
```

class AbsLimitCls

AbsLimit commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthE=BandwidthE.Default*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:VHTofdm:BW
↪<bandwidth>:ABSLimit
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.vht0fdm.bw.
↪absLimit.get(bandwidthE = repcap.BandwidthE.Default)
```

Defines an absolute power limit for 802.11ac signals with the specified <bandwidth>. See 'Transmit spectrum mask OFDM, absolute limits' for background information.

param bandwidthE
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

return
tsm_lim_abs: Limit value applies to frequency offsets greater than 3/2*bandwidth, measured at 100 kHz RBW.

set(*tsm_lim_abs: float*, *bandwidthE=BandwidthE.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:VHTofdm:BW
↪<bandwidth>:ABSLimit
driver.configure.wlanMeas.multiEval.limit.tsMask.vhtOfdm.bw.absLimit.set(tsm_
↪lim_abs = 1.0, bandwidthE = repcap.BandwidthE.Default)
```

Defines an absolute power limit for 802.11ac signals with the specified <bandwidth>. See 'Transmit spectrum mask OFDM, absolute limits' for background information.

param tsm_lim_abs
Limit value applies to frequency offsets greater than 3/2*bandwidth, measured at 100 kHz RBW.

param bandwidthE
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

6.1.1.3.3.118 Enable

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:VHTofdm:BW<bandwidth>
↪:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthE=BandwidthE.Default*) → bool

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:VHTofdm:BW
↪<bandwidth>:ENABle
value: bool = driver.configure.wlanMeas.multiEval.limit.tsMask.vhtOfdm.bw.
↪enable.get(bandwidthE = repcap.BandwidthE.Default)
```

Enables or disables the transmit spectrum mask for 802.11ac signals with the specified <bandwidth>, i.e. activates or deactivates the corresponding limit checks.

param bandwidthE
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

return
tsm_lim_enable: No help available

set(*tsm_lim_enable: bool*, *bandwidthE=BandwidthE.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:VHTofdm:BW
↪<bandwidth>:ENABle
driver.configure.wlanMeas.multiEval.limit.tsMask.vhtOfdm.bw.enable.set(tsm_lim_
↪enable = False, bandwidthE = repcap.BandwidthE.Default)
```

Enables or disables the transmit spectrum mask for 802.11ac signals with the specified <bandwidth>, i.e. activates or deactivates the corresponding limit checks.

param tsm_lim_enable
No help available

param bandwidthE
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

6.1.1.3.3.119 Y

class YCls

Y commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.limit.tsMask.vhtOfdm.bw.y.clone()
```

Subgroups

6.1.1.3.3.120 A

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:VHTofdm:BW<bandwidth>:Y:A
```

class ACls

A commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthE=BandwidthE.Default*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:VHTofdm:BW
↪<bandwidth>:Y:A
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.vhtOfdm.bw.y.a.
↪get(bandwidthE = repcap.BandwidthE.Default)
```

Defines the relative spectral density limit for point A (frequency offset: 2*bandwidth) on the transmit spectrum mask for 802.11ac signals with the specified <bandwidth>. See 'Transmit spectrum mask OFDM, default masks' for background information.

param bandwidthE
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

return
tsm_lim_yrel_lev_a: No help available

set(*tsm_lim_yrel_lev_a: float*, *bandwidthE=BandwidthE.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:VHTofdm:BW
↪<bandwidth>:Y:A
driver.configure.wlanMeas.multiEval.limit.tsMask.vhtOfdm.bw.y.a.set(tsm_lim_
↪yrel_lev_a = 1.0, bandwidthE = repcap.BandwidthE.Default)
```

Defines the relative spectral density limit for point A (frequency offset: 2*bandwidth) on the transmit spectrum mask for 802.11ac signals with the specified <bandwidth>. See 'Transmit spectrum mask OFDM, default masks' for background information.

param tsm_lim_yrel_lev_a
No help available

param bandwidthE
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

6.1.1.3.3.121 B

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:VHTofdm:BW<bandwidth>:Y:B
```

class BCls

B commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthE=BandwidthE.Default*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:VHTofdm:BW
↪<bandwidth>:Y:B
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.vhtOfdm.bw.y.b.
↪get(bandwidthE = repcap.BandwidthE.Default)
```

Defines the relative spectral density limit for point B (frequency offset: 3/2*bandwidth) on the transmit spectrum mask for 802.11ac signals with the specified <bandwidth>. See 'Transmit spectrum mask OFDM, default masks' for background information.

param bandwidthE
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

return
tsm_lim_yrel_lev_b: No help available

set(*tsm_lim_yrel_lev_b: float*, *bandwidthE=BandwidthE.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:VHTofdm:BW
↪<bandwidth>:Y:B
driver.configure.wlanMeas.multiEval.limit.tsMask.vhtOfdm.bw.y.b.set(tsm_lim_
↪yrel_lev_b = 1.0, bandwidthE = repcap.BandwidthE.Default)
```

Defines the relative spectral density limit for point B (frequency offset: 3/2*bandwidth) on the transmit spectrum mask for 802.11ac signals with the specified <bandwidth>. See 'Transmit spectrum mask OFDM, default masks' for background information.

param tsm_lim_yrel_lev_b
No help available

param bandwidthE
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

6.1.1.3.3.122 C

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:VHTofdm:BW<bandwidth>:Y:C
```

class CCls

C commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthE=BandwidthE.Default*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:VHTofdm:BW
↪<bandwidth>:Y:C
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.vhtOfdm.bw.y.c.
↪get(bandwidthE = repcap.BandwidthE.Default)
```

Defines the relative spectral density limit for point C (frequency offset: 1*bandwidth) on the transmit spectrum mask for 802.11ac signals with the specified <bandwidth>. See 'Transmit spectrum mask OFDM, default masks' for background information.

param bandwidthE
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

return
tsm_lim_yrel_lev_c: No help available

set(*tsm_lim_yrel_lev_c: float*, *bandwidthE=BandwidthE.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:VHTofdm:BW
↪<bandwidth>:Y:C
driver.configure.wlanMeas.multiEval.limit.tsMask.vhtOfdm.bw.y.c.set(tsm_lim_
↪yrel_lev_c = 1.0, bandwidthE = repcap.BandwidthE.Default)
```

Defines the relative spectral density limit for point C (frequency offset: 1*bandwidth) on the transmit spectrum mask for 802.11ac signals with the specified <bandwidth>. See 'Transmit spectrum mask OFDM, default masks' for background information.

param tsm_lim_yrel_lev_c
No help available

param bandwidthE
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

6.1.1.3.3.123 D

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:VHTofdm:BW<bandwidth>:Y:D
```

class DCls

D commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*bandwidthE=BandwidthE.Default*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:VHTofdm:BW
↪<bandwidth>:Y:D
value: float = driver.configure.wlanMeas.multiEval.limit.tsMask.vhtOfdm.bw.y.d.
↪get(bandwidthE = repcap.BandwidthE.Default)
```

Defines the relative spectral density limit for point D (center frequency offset: 1/2*bandwidth + 1 MHz) on the transmit spectrum mask for 802.11ac signals with the specified <bandwidth>. See 'Transmit spectrum mask OFDM, default masks' for background information.

param bandwidthE
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

return
tsm_lim_yrel_lev_d: No help available

set(*tsm_lim_yrel_lev_d: float*, *bandwidthE=BandwidthE.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:VHTofdm:BW
↪<bandwidth>:Y:D
driver.configure.wlanMeas.multiEval.limit.tsMask.vhtOfdm.bw.y.d.set(tsm_lim_
↪yrel_lev_d = 1.0, bandwidthE = repcap.BandwidthE.Default)
```

Defines the relative spectral density limit for point D (center frequency offset: 1/2*bandwidth + 1 MHz) on the transmit spectrum mask for 802.11ac signals with the specified <bandwidth>. See 'Transmit spectrum mask OFDM, default masks' for background information.

param tsm_lim_yrel_lev_d
No help available

param bandwidthE
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Bw')

6.1.1.3.4 ListPy

SCPI Commands :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:COUNt
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:CMODe
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:STIMe
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:MTIMe
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:MOFFset
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:ENPower
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:FREQuency
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:STANdard
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:BWIDth
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:BTYPe
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:RTRigger
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST
```

class ListPyCls

ListPy commands group definition. 30 total commands, 4 Subgroups, 12 group commands

get_bandwidth() → List[Bandwidth]

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:BWIDth
value: List[enums.Bandwidth] = driver.configure.wlanMeas.multiEval.listPy.get_
↪bandwidth()
```

Specifies the channel bandwidths for all segments in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

return
bandwidths: No help available

get_btype() → List[BurstTypeB]

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:BTYPe
value: List[enums.BurstTypeB] = driver.configure.wlanMeas.multiEval.listPy.get_
↪btype()
```

Specifies the burst types for standard 802.11n for all segments in list mode. Do not use the command for other standards. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

return
burst_types: No help available

get_cmode() → ParameterSetMode

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:CMODe
value: enums.ParameterSetMode = driver.configure.wlanMeas.multiEval.listPy.get_
↪cmode()
```

Specifies how the input connector is selected for list mode measurements.

return
connector_mode: GLOBal: Use the same RF connection for all segments. See ROUTe:WLAN:MEASi:SPATh LIST: Assign a connection to each segment. CONFigure:WLAN:MEASi:MEValuation:LIST:SEGMentno:CIDX

get_count() → int

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:COUNt
value: int = driver.configure.wlanMeas.multiEval.listPy.get_count()
```

Defines the number of segments in the entire measurement interval.

return
no_of_segments: No help available

get_envelope_power() → List[float]

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:ENPower
value: List[float] = driver.configure.wlanMeas.multiEval.listPy.get_envelope_
↪power()
```

Specifies the expected nominal power of the measured RF signal for all segments in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1,

{...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

return
levels: No help available

get_frequency() → List[float]

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:FREQuency
value: List[float] = driver.configure.wlanMeas.multiEval.listPy.get_frequency()
```

Specifies the measurement frequencies for all segments in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

return
frequencies: No help available

get_moffset() → List[float]

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:MOFFset
value: List[float] = driver.configure.wlanMeas.multiEval.listPy.get_moffset()
```

Specifies the measurement offsets for all segments in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

return
meas_offsets: No help available

get_mtime() → List[float]

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:MTIMe
value: List[float] = driver.configure.wlanMeas.multiEval.listPy.get_mtime()
```

Specifies the measurement times for all segments in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

return
meas_times: No help available

get_rtrigger() → List[bool]

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:RTRigger
value: List[bool] = driver.configure.wlanMeas.multiEval.listPy.get_rtrigger()
```

Specifies whether the measurement in list mode waits for a trigger event before measuring the segment, or not. For the first segment, the value OFF is always interpreted as ON. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas. Configure.WlanMeas.MultiEval.ListPy.count.

return
retriggers: No help available

get_standard() → List[IeeeStandard]

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:STANdard
value: List[enums.IeeeStandard] = driver.configure.wlanMeas.multiEval.listPy.
↪get_standard()
```

Specifies the standard for all segments in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas. Configure.WlanMeas.MultiEval.ListPy.count.

return
standards: No help available

get_stime() → List[float]

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:STIMe
value: List[float] = driver.configure.wlanMeas.multiEval.listPy.get_stime()
```

Specifies the segment times for all segments in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas. Configure.WlanMeas.MultiEval.ListPy.count.

return
segment_times: No help available

get_value() → bool

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST
value: bool = driver.configure.wlanMeas.multiEval.listPy.get_value()
```

Enables or disables the list mode.

return
list_mode_enable: OFF: Disable list mode. ON: Enable list mode.

set_bandwidth(*bandwidths: List[Bandwidth]*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:BWIDth
driver.configure.wlanMeas.multiEval.listPy.set_bandwidth(bandwidths =␣
↪[Bandwidth.BW05mhz, Bandwidth.BW88mhz])
```

Specifies the channel bandwidths for all segments in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

param bandwidths
No help available

set_btype(*burst_types: List[BurstTypeB]*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:BTYPe
driver.configure.wlanMeas.multiEval.listPy.set_btype(burst_types = [BurstTypeB.
↪GREenfield, BurstTypeB.MIXed])
```

Specifies the burst types for standard 802.11n for all segments in list mode. Do not use the command for other standards. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

param burst_types
No help available

set_cmode(*connector_mode: ParameterSetMode*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:CMODe
driver.configure.wlanMeas.multiEval.listPy.set_cmode(connector_mode = enums.
↪ParameterSetMode.GLOBal)
```

Specifies how the input connector is selected for list mode measurements.

param connector_mode
GLOBal: Use the same RF connection for all segments. See ROUTe:WLAN:MEASi:SPATh LIST: Assign a connection to each segment. CONFigure:WLAN:MEASi:MEValuation:LIST:SEGMentno:CIDX

set_count(*no_of_segments: int*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:COUNt
driver.configure.wlanMeas.multiEval.listPy.set_count(no_of_segments = 1)
```

Defines the number of segments in the entire measurement interval.

param no_of_segments
No help available

set_envelope_power(*levels: List[float]*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:ENPower
driver.configure.wlanMeas.multiEval.listPy.set_envelope_power(levels = [1.1, 2.
↪2, 3.3])
```

Specifies the expected nominal power of the measured RF signal for all segments in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

param levels
No help available

set_frequency(*frequencies: List[float]*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:FREQuency
driver.configure.wlanMeas.multiEval.listPy.set_frequency(frequencies = [1.1, 2.
↪2, 3.3])
```

Specifies the measurement frequencies for all segments in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

param frequencies
No help available

set_moffset(*meas_offsets: List[float]*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:MOFFset
driver.configure.wlanMeas.multiEval.listPy.set_moffset(meas_offsets = [1.1, 2.2,
↪ 3.3])
```

Specifies the measurement offsets for all segments in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

param meas_offsets
No help available

set_mtime(*meas_times: List[float]*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:MTIMe
driver.configure.wlanMeas.multiEval.listPy.set_mtime(meas_times = [1.1, 2.2, 3.
↪3])
```

Specifies the measurement times for all segments in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

param meas_times
No help available

set_rtrigger(*retriggers: List[bool]*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:RTRigger
driver.configure.wlanMeas.multiEval.listPy.set_rtrigger(retriggers = [True,␣
↪False, True])
```

Specifies whether the measurement in list mode waits for a trigger event before measuring the segment, or not. For the first segment, the value OFF is always interpreted as ON. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas. Configure.WlanMeas.MultiEval.ListPy.count.

param retriggers
No help available

set_standard(*standards: List[IeeeStandard]*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:STANdard
driver.configure.wlanMeas.multiEval.listPy.set_standard(standards =␣
↪[IeeeStandard.DSSS, IeeeStandard.VHTofdm])
```

Specifies the standard for all segments in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas. Configure.WlanMeas.MultiEval.ListPy.count.

param standards
No help available

set_stime(*segment_times: List[float]*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:STIMe
driver.configure.wlanMeas.multiEval.listPy.set_stime(segment_times = [1.1, 2.2,␣
↪3.3])
```

Specifies the segment times for all segments in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas. Configure.WlanMeas.MultiEval.ListPy.count.

param segment_times
No help available

set_value(*list_mode_enable: bool*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST
driver.configure.wlanMeas.multiEval.listPy.set_value(list_mode_enable = False)
```

Enables or disables the list mode.

param list_mode_enable
OFF: Disable list mode. ON: Enable list mode.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.listPy.clone()
```

Subgroups

6.1.1.3.4.1 Result

SCPI Commands :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:RESult:MODulation
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:RESult:TSMask
```

class ResultCls

Result commands group definition. 2 total commands, 0 Subgroups, 2 group commands

get_modulation() → List[bool]

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:RESult:MODulation
value: List[bool] = driver.configure.wlanMeas.multiEval.listPy.result.get_
↪modulation()
```

Enables or disables the evaluation of results for modulation (..:MODulation) and transmit spectrum mask (.. :TSMask) measurements in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {.. .}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas. MultiEval.ListPy.count.

return
enable_mod: No help available

get_ts_mask() → List[bool]

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:RESult:TSMask
value: List[bool] = driver.configure.wlanMeas.multiEval.listPy.result.get_ts_
↪mask()
```

Enables or disables the evaluation of results for modulation (..:MODulation) and transmit spectrum mask (.. :TSMask) measurements in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {.. .}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas. MultiEval.ListPy.count.

return
enable_sem: No help available

set_modulation(*enable_mod: List[bool]*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:RESult:MODulation
driver.configure.wlanMeas.multiEval.listPy.result.set_modulation(enable_mod =␣
↪[True, False, True])
```

Enables or disables the evaluation of results for modulation (..:MODulation) and transmit spectrum mask (.. :TSMask) measurements in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {.. .}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas. MultiEval.ListPy.count.

param enable_mod
No help available

set_ts_mask(*enable_sem: List[bool]*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:RESult:TSMask
driver.configure.wlanMeas.multiEval.listPy.result.set_ts_mask(enable_sem =␣
↪[True, False, True])
```

Enables or disables the evaluation of results for modulation (..:MODulation) and transmit spectrum mask (.. :TSMask) measurements in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {.. .}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas. MultiEval.ListPy.count.

param enable_sem
No help available

6.1.1.3.4.2 Scount

SCPI Commands :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SCOunt:MODulation
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SCOunt:TSMask
```

class ScountCls

Scount commands group definition. 2 total commands, 0 Subgroups, 2 group commands

get_modulation() → List[int]

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SCOunt:MODulation
value: List[int] = driver.configure.wlanMeas.multiEval.listPy.scount.get_
↪modulation()
```

Specifies the statistical length for modulation measurements for all segments in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

return
stat_counts_mod: No help available

get_ts_mask() → List[int]

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SCOunt:TSMask
value: List[int] = driver.configure.wlanMeas.multiEval.listPy.scount.get_ts_
↪mask()
```

Specifies the spectrum statistical length for transmit spectrum mask measurements for all segments in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

return
stat_counts_sem: No help available

set_modulation(*stat_counts_mod: List[int]*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SCOunt:MODulation
driver.configure.wlanMeas.multiEval.listPy.scount.set_modulation(stat_counts_
↪mod = [1, 2, 3])
```

Specifies the statistical length for modulation measurements for all segments in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

param stat_counts_mod
No help available

set_ts_mask(*stat_counts_sem: List[int]*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SCOunt:TSMask
driver.configure.wlanMeas.multiEval.listPy.scount.set_ts_mask(stat_counts_sem =␣
↪[1, 2, 3])
```

Specifies the spectrum statistical length for transmit spectrum mask measurements for all segments in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

param stat_counts_sem
No help available

6.1.1.3.4.3 Segment<SegmentB>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.configure.wlanMeas.multiEval.listPy.segment.repcap_segmentB_get()
driver.configure.wlanMeas.multiEval.listPy.segment.repcap_segmentB_set(repcap.SegmentB.
↪Nr1)
```

class SegmentCls

Segment commands group definition. 13 total commands, 13 Subgroups, 0 group commands Repeated Capability: SegmentB, default value after init: SegmentB.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.listPy.segment.clone()
```

Subgroups

6.1.1.3.4.4 Bandwidth

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>:BWIDth
```

class BandwidthCls

Bandwidth commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*segmentB=SegmentB.Default*) → Bandwidth

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:BWIDth
value: enums.Bandwidth = driver.configure.wlanMeas.multiEval.listPy.segment.
↪bandwidth.get(segmentB = repcap.SegmentB.Default)
```

Specifies the channel bandwidth for segment <no> in list mode.

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
bandwidth: BW05mhz: 5 MHz BW10mhz: 10 MHz BW20mhz: 20 MHz BW40mhz: 40 MHz BW80mhz: 80 MHz BW16mhz: 160 MHz BW32mhz: 320 MHz

set(*bandwidth: Bandwidth*, *segmentB=SegmentB.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:BWIDth
driver.configure.wlanMeas.multiEval.listPy.segment.bandwidth.set(bandwidth =␣
↪enums.Bandwidth.BW05mhz, segmentB = repcap.SegmentB.Default)
```

Specifies the channel bandwidth for segment <no> in list mode.

param bandwidth
BW05mhz: 5 MHz BW10mhz: 10 MHz BW20mhz: 20 MHz BW40mhz: 40 MHz BW80mhz: 80 MHz BW16mhz: 160 MHz BW32mhz: 320 MHz

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

6.1.1.3.4.5 Btype

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>:BTYPe
```

class BtypeCls

Btype commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*segmentB=SegmentB.Default*) → BurstTypeB

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:BTYPe
value: enums.BurstTypeB = driver.configure.wlanMeas.multiEval.listPy.segment.
↪btype.get(segmentB = repcap.SegmentB.Default)
```

Specifies the burst types for standard 802.11n for segment <no> in list mode. Do not use the command for other standards.

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
burst_type: MIXed: for coexistence with other standards GREenfield: incompatible with other standards

set(*burst_type: BurstTypeB*, *segmentB=SegmentB.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:BTYPe
driver.configure.wlanMeas.multiEval.listPy.segment.btype.set(burst_type = enums.
↪BurstTypeB.GREenfield, segmentB = repcap.SegmentB.Default)
```

Specifies the burst types for standard 802.11n for segment <no> in list mode. Do not use the command for other standards.

param burst_type
MIXed: for coexistence with other standards GREenfield: incompatible with other standards

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

6.1.1.3.4.6 EnvelopePower

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>:ENPower
```

class EnvelopePowerCls

EnvelopePower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*segmentB=SegmentB.Default*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:ENPower
value: float = driver.configure.wlanMeas.multiEval.listPy.segment.envelopePower.
↪get(segmentB = repcap.SegmentB.Default)
```

Specifies the expected nominal power of the measured RF signal for segment <no> in list mode.

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
level: The range of the expected nominal power can be calculated as follows: Range (Expected Nominal Power) = Range (Input Power) + External Attenuation - User Margin

set(*level: float*, *segmentB=SegmentB.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:ENPower
driver.configure.wlanMeas.multiEval.listPy.segment.envelopePower.set(level = 1.
↪0, segmentB = repcap.SegmentB.Default)
```

Specifies the expected nominal power of the measured RF signal for segment <no> in list mode.

param level
The range of the expected nominal power can be calculated as follows: Range (Expected Nominal Power) = Range (Input Power) + External Attenuation - User Margin

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

6.1.1.3.4.7 Frequency

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>:FREQuency
```

class FrequencyCls

Frequency commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*segmentB=SegmentB.Default*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:FREQuency
value: float = driver.configure.wlanMeas.multiEval.listPy.segment.frequency.
↪get(segmentB = repcap.SegmentB.Default)
```

Specifies the center frequency of the RF analyzer for segment <no> in list mode. For the supported frequency range, see 'Frequency ranges'.

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
frequency: No help available

set(*frequency: float*, *segmentB=SegmentB.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:FREQuency
driver.configure.wlanMeas.multiEval.listPy.segment.frequency.set(frequency = 1.
↪0, segmentB = repcap.SegmentB.Default)
```

Specifies the center frequency of the RF analyzer for segment <no> in list mode. For the supported frequency range, see 'Frequency ranges'.

param frequency
No help available

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

6.1.1.3.4.8 Moffset

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>:MOFFset
```

class MoffsetCls

Moffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*segmentB=SegmentB.Default*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:MOFFset
value: float = driver.configure.wlanMeas.multiEval.listPy.segment.moffset.
↪get(segmentB = repcap.SegmentB.Default)
```

Specifies the measurement offset for segment <no> in list mode.

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
meas_offset: No help available

set(*meas_offset: float*, *segmentB=SegmentB.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:MOFFset
driver.configure.wlanMeas.multiEval.listPy.segment.moffset.set(meas_offset = 1.
↪0, segmentB = repcap.SegmentB.Default)
```

Specifies the measurement offset for segment <no> in list mode.

param meas_offset
No help available

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

6.1.1.3.4.9 Mtime

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>:MTIMe
```

class MtimeCls

Mtime commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*segmentB=SegmentB.Default*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:MTIMe
value: float = driver.configure.wlanMeas.multiEval.listPy.segment.mtime.
↪get(segmentB = repcap.SegmentB.Default)
```

Specifies the measurement time for segment <no> in list mode.

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
meas_time: Duration of measurement for the segment

set(*meas_time: float*, *segmentB=SegmentB.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:MTIMe
driver.configure.wlanMeas.multiEval.listPy.segment.mtime.set(meas_time = 1.0,␣
↪segmentB = repcap.SegmentB.Default)
```

Specifies the measurement time for segment <no> in list mode.

param meas_time
Duration of measurement for the segment

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

6.1.1.3.4.10 Result

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class ResultStruct

Response structure. Fields:

- Enable_Mod: bool: No parameter help available
- Enable_Sem: bool: No parameter help available

get(*segmentB=SegmentB.Default*) → ResultStruct

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:RESult
value: ResultStruct = driver.configure.wlanMeas.multiEval.listPy.segment.result.
↪get(segmentB = repcap.SegmentB.Default)
```

Enables or disables the evaluation of results for modulation and transmit spectrum mask measurements for segment <no> in list mode.

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
structure: for return value, see the help for ResultStruct structure arguments.

set(*enable_mod: bool*, *enable_sem: bool*, *segmentB=SegmentB.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:RESult
driver.configure.wlanMeas.multiEval.listPy.segment.result.set(enable_mod =␣
↪False, enable_sem = False, segmentB = repcap.SegmentB.Default)
```

Enables or disables the evaluation of results for modulation and transmit spectrum mask measurements for segment <no> in list mode.

param enable_mod
No help available

param enable_sem
No help available

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

6.1.1.3.4.11 Rtrigger

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>:RTRigger
```

class RtriggerCls

Rtrigger commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*segmentB=SegmentB.Default*) → bool

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:RTRigger
value: bool = driver.configure.wlanMeas.multiEval.listPy.segment.rtrigger.
↪get(segmentB = repcap.SegmentB.Default)
```

Specifies for segment <no> in list mode, whether the measurement waits for a trigger event before measuring the segment, or not. For the first segment, the value OFF is always interpreted as ON. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
retrigger: OFF: measure the segment without retrigger ON: wait for a trigger event from the trigger source configured via method RsCMPX_WlanMeas.Trigger.WlanMeas.MultiEval.source

set(*retrigger: bool*, *segmentB=SegmentB.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:RTRigger
driver.configure.wlanMeas.multiEval.listPy.segment.rtrigger.set(retrigger =␣
↪False, segmentB = repcap.SegmentB.Default)
```

Specifies for segment <no> in list mode, whether the measurement waits for a trigger event before measuring the segment, or not. For the first segment, the value OFF is always interpreted as ON. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

param retrigger
OFF: measure the segment without retrigger ON: wait for a trigger event from the trigger source configured via method RsCMPX_WlanMeas.Trigger.WlanMeas.MultiEval.source

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

6.1.1.3.4.12 Scount

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>:SCOunt
```

class ScountCls

Scount commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class ScountStruct

Response structure. Fields:

- Stat_Count_Mod: int: No. of burst to be measured during modulation measurements.
- Stat_Count_Sem: int: No. of bursts to be measured during spectrum measurements.

get(*segmentB=SegmentB.Default*) → ScountStruct

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:SCOunt
value: ScountStruct = driver.configure.wlanMeas.multiEval.listPy.segment.scount.
↪get(segmentB = repcap.SegmentB.Default)
```

Specifies the modulation and spectrum statistical length for segment <no> in list mode.

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
structure: for return value, see the help for ScountStruct structure arguments.

set(*stat_count_mod: int*, *stat_count_sem: int*, *segmentB=SegmentB.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:SCOunt
driver.configure.wlanMeas.multiEval.listPy.segment.scount.set(stat_count_mod =␣
↪1, stat_count_sem = 1, segmentB = repcap.SegmentB.Default)
```

Specifies the modulation and spectrum statistical length for segment <no> in list mode.

param stat_count_mod
No. of burst to be measured during modulation measurements.

param stat_count_sem
No. of bursts to be measured during spectrum measurements.

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

6.1.1.3.4.13 Setup

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>:SETup
```

class SetupCls

Setup commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class SetupStruct

Structure for setting input parameters. Fields:

- Segment_Time: float: Duration of the segment
- Meas_Time: float: Duration of measurement for the segment
- Meas_Offset: float: Measurement offset for the segment
- Level: float: Expected nominal power of the measured RF signal within the segment The range of the expected nominal power can be calculated as follows: Range (Expected Nominal Power) = Range (Input Power) + External Attenuation - User Margin
- Frequency: float: Configures the center frequency of the RF analyzer. Set it to the center frequency of the received WLAN channel.
- Standard: enums.IeeeStandard: DSSS: 802.11b/g (DSSS) LOFDm: 802.11a/g (OFDM) HTOFdm: 802.11n VHTofdm: 802.11ac EHTofdm: 802.11be
- Bandwidth: enums.Bandwidth: BW05mhz: 5 MHz BW10mhz: 10 MHz BW20mhz: 20 MHz BW40mhz: 40 MHz BW80mhz: 80 MHz BW16mhz: 160 MHz BW32mhz: 320 MHz

get(*segmentB=SegmentB.Default*) → SetupStruct

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:SETup
value: SetupStruct = driver.configure.wlanMeas.multiEval.listPy.segment.setup.
↪get(segmentB = repcap.SegmentB.Default)
```

Specifies burst parameter settings for segment <no> in list mode. Send this command for all segments to be measured.

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
structure: for return value, see the help for SetupStruct structure arguments.

set(*structure: SetupStruct*, *segmentB=SegmentB.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:SETup
structure = driver.configure.wlanMeas.multiEval.listPy.segment.setup.
↪SetupStruct()
structure.Segment_Time: float = 1.0
structure.Meas_Time: float = 1.0
structure.Meas_Offset: float = 1.0
structure.Level: float = 1.0
```

(continues on next page)

(continued from previous page)

```
structure.Frequency: float = 1.0
structure.Standard: enums.IeeeStandard = enums.IeeeStandard.DSSS
structure.Bandwidth: enums.Bandwidth = enums.Bandwidth.BW05mhz
driver.configure.wlanMeas.multiEval.listPy.segment.setup.set(structure,␣
↪segmentB = repcap.SegmentB.Default)
```

Specifies burst parameter settings for segment <no> in list mode. Send this command for all segments to be measured.

param structure
for set value, see the help for SetupStruct structure arguments.

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

6.1.1.3.4.14 SingleCmw

class SingleCmwCls

SingleCmw commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.multiEval.listPy.segment.singleCmw.clone()
```

Subgroups

6.1.1.3.4.15 Connector

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>:CMWS:CONNector
```

class ConnectorCls

Connector commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*segmentB=SegmentB.Default*) → ConnectorSwitchExt

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:CMWS:CONNector
value: enums.ConnectorSwitchExt = driver.configure.wlanMeas.multiEval.listPy.
↪segment.singleCmw.connector.get(segmentB = repcap.SegmentB.Default)
```

No command help available

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
connector: No help available

set(*connector: ConnectorSwitchExt*, *segmentB=SegmentB.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:CMWS:CONNector
driver.configure.wlanMeas.multiEval.listPy.segment.singleCmw.connector.
↪set(connector = enums.ConnectorSwitchExt.OFF, segmentB = repcap.SegmentB.
↪Default)
```

No command help available

param connector
No help available

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

6.1.1.3.4.16 Standard

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>:STANdard
```

class StandardCls

Standard commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*segmentB=SegmentB.Default*) → IeeeStandard

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:STANdard
value: enums.IeeeStandard = driver.configure.wlanMeas.multiEval.listPy.segment.
↪standard.get(segmentB = repcap.SegmentB.Default)
```

Specifies the standard for segment <no> in list mode.

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
standard: DSSS: 802.11b/g (DSSS) LOFDm: 802.11a/g (OFDM) HTOFdm: 802.11n VHTofdm: 802.11ac EHTofdm: 802.11be

set(*standard: IeeeStandard*, *segmentB=SegmentB.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:STANdard
driver.configure.wlanMeas.multiEval.listPy.segment.standard.set(standard =␣
↪enums.IeeeStandard.DSSS, segmentB = repcap.SegmentB.Default)
```

Specifies the standard for segment <no> in list mode.

param standard
DSSS: 802.11b/g (DSSS) LOFDm: 802.11a/g (OFDM) HTOFdm: 802.11n VHTofdm: 802.11ac EHTofdm: 802.11be

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

6.1.1.3.4.17 Stime

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>:STIMe
```

class StimeCls

Stime commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*segmentB=SegmentB.Default*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:STIMe
value: float = driver.configure.wlanMeas.multiEval.listPy.segment.stime.
↪get(segmentB = repcap.SegmentB.Default)
```

Specifies the duration of the segment for segment <no> in list mode.

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
segment_time: No help available

set(*segment_time: float*, *segmentB=SegmentB.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:STIMe
driver.configure.wlanMeas.multiEval.listPy.segment.stime.set(segment_time = 1.0,
↪ segmentB = repcap.SegmentB.Default)
```

Specifies the duration of the segment for segment <no> in list mode.

param segment_time
No help available

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

6.1.1.3.4.18 SingleCmw

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:CMWS:CONNector
```

class SingleCmwCls

SingleCmw commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_connector() → List[ConnectorSwitchExt]

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:CMWS:CONNector
value: List[enums.ConnectorSwitchExt] = driver.configure.wlanMeas.multiEval.
↪listPy.singleCmw.get_connector()
```

No command help available

return
connectors: No help available

set_connector(*connectors: List[ConnectorSwitchExt]*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:CMWS:CONNector
driver.configure.wlanMeas.multiEval.listPy.singleCmw.set_connector(connectors =␣
↪[ConnectorSwitchExt.OFF, ConnectorSwitchExt.RH8])
```

No command help available

param connectors
No help available

6.1.1.3.5 PowerVsTime

SCPI Commands :

```
CONFigure:WLAN:MEASurement<instance>:MEValuation:PVTime:RPOWer
CONFigure:WLAN:MEASurement<instance>:MEValuation:PVTime:ALENgth
CONFigure:WLAN:MEASurement<Instance>:MEValuation:PVTime:REDGe
CONFigure:WLAN:MEASurement<Instance>:MEValuation:PVTime:FEDGe
CONFigure:WLAN:MEASurement<Instance>:MEValuation:PVTime:BURSt
```

class PowerVsTimeCls

PowerVsTime commands group definition. 5 total commands, 0 Subgroups, 5 group commands

get_alength() → float

```
# SCPI: CONFigure:WLAN:MEASurement<instance>:MEValuation:PVTime:ALENgth
value: float = driver.configure.wlanMeas.multiEval.powerVsTime.get_alength()
```

Sets the length of the moving average filter, which smoothes the power trace and thus eliminates the modulation.

return
avg_lenth: No help available

get_burst() → bool

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:PVTime:BURSt
value: bool = driver.configure.wlanMeas.multiEval.powerVsTime.get_burst()
```

Enables or disables the evaluation of burst power results in the power vs time square.

return
burst: OFF: Do not evaluate results. ON: Evaluate the results.

get_falling_edge() → bool

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:PVTime:FEDGe
value: bool = driver.configure.wlanMeas.multiEval.powerVsTime.get_falling_edge()
```

Enables or disables the evaluation of falling edge results (transmit power-down ramp) in the power vs time square for DSSS signals.

return
fall: OFF: Do not evaluate results. ON: Evaluate the results.

get_rising_edge() → bool

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:PVTime:REDGe
value: bool = driver.configure.wlanMeas.multiEval.powerVsTime.get_rising_edge()
```

Enables or disables the evaluation of rising edge results (transmit power-on ramp) in the power vs time square for DSSS signals.

return
rising: OFF: Do not evaluate results. ON: Evaluate the results.

get_rpower() → RefPower

```
# SCPI: CONFigure:WLAN:MEASurement<instance>:MEValuation:PVTime:RPOWer
value: enums.RefPower = driver.configure.wlanMeas.multiEval.powerVsTime.get_
↪rpower()
```

Sets the reference power to the maximum power or to the mean power of the burst. In DSSS, the thresholds for rising and falling edge results are defined as percentage values of the reference power.

return
ref_power: No help available

set_alength(*avg_lenth: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<instance>:MEValuation:PVTime:ALENgth
driver.configure.wlanMeas.multiEval.powerVsTime.set_alength(avg_lenth = 1.0)
```

Sets the length of the moving average filter, which smoothes the power trace and thus eliminates the modulation.

param avg_lenth
No help available

set_burst(*burst: bool*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:PVTime:BURSt
driver.configure.wlanMeas.multiEval.powerVsTime.set_burst(burst = False)
```

Enables or disables the evaluation of burst power results in the power vs time square.

param burst
OFF: Do not evaluate results. ON: Evaluate the results.

set_falling_edge(*fall: bool*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:PVTime:FEDGe
driver.configure.wlanMeas.multiEval.powerVsTime.set_falling_edge(fall = False)
```

Enables or disables the evaluation of falling edge results (transmit power-down ramp) in the power vs time square for DSSS signals.

param fall
OFF: Do not evaluate results. ON: Evaluate the results.

set_rising_edge(*rising: bool*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:PVTime:REDGe
driver.configure.wlanMeas.multiEval.powerVsTime.set_rising_edge(rising = False)
```

Enables or disables the evaluation of rising edge results (transmit power-on ramp) in the power vs time square for DSSS signals.

param rising
OFF: Do not evaluate results. ON: Evaluate the results.

set_rpower(*ref_power: RefPower*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<instance>:MEValuation:PVTime:RPOWer
driver.configure.wlanMeas.multiEval.powerVsTime.set_rpower(ref_power = enums.
↪RefPower.MAXimum)
```

Sets the reference power to the maximum power or to the mean power of the burst. In DSSS, the thresholds for rising and falling edge results are defined as percentage values of the reference power.

param ref_power
No help available

6.1.1.3.6 Result

SCPI Commands :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:PVTime
CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:SFLatness
CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult[:ALL]
CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:EVM
CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:EVMCarrier
CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:IQConst
CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:UTERror
CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:EVMSymbol
CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:TSMask
CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:MSCalar
```

class ResultCls

Result commands group definition. 10 total commands, 0 Subgroups, 10 group commands

class AllStruct

Structure for setting input parameters. Contains optional set arguments. Fields:

- Mod_Scalar: bool: Modulation scalar overview OFF: Do not evaluate results. ON: Evaluate the results.
- Power_Vs_Time: bool: Power vs time
- Evm_Vs_Chip: bool: EVM vs chip
- Evm_Vs_Sym: bool: EVM vs symbol
- Evm_Vs_Carr: bool: EVM vs carrier
- Iq_Const: bool: I/Q constellation diagram
- Spec_Flatness: bool: Spectrum flatness
- Tran_Spec_Mask: bool: Transmit spectrum mask.
- Unused_Tone_Err: bool: Optional setting parameter. Unused tone error

get_all() → AllStruct

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult[:ALL]
value: AllStruct = driver.configure.wlanMeas.multiEval.result.get_all()
```

Enables or disables the evaluation of results in the multi-evaluation measurement. This command combines all other CONFigure:WLAN:MEAS<i>:MEValuation:RESult... commands.

return
structure: for return value, see the help for AllStruct structure arguments.

get_evm() → bool

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:EVM
value: bool = driver.configure.wlanMeas.multiEval.result.get_evm()
```

Enables or disables the evaluation of EVM vs chip results.

return
evm_enable: OFF: Do not evaluate results. ON: Evaluate the results.

get_evm_carrier() → bool

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:EVMCarrier
value: bool = driver.configure.wlanMeas.multiEval.result.get_evm_carrier()
```

Enables or disables the evaluation of EVM vs carrier results.

return
evm_enable: OFF: Do not evaluate results. ON: Evaluate the results.

get_evm_symbol() → bool

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:EVMSymbol
value: bool = driver.configure.wlanMeas.multiEval.result.get_evm_symbol()
```

Enables or disables the evaluation of EVM vs symbol results.

return
evm_enable: OFF: Do not evaluate results. ON: Evaluate the results.

get_iq_constant() → bool

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:IQConst
value: bool = driver.configure.wlanMeas.multiEval.result.get_iq_constant()
```

Enables or disables the evaluation of I/Q constellation results.

return
iq_enable: OFF: Do not evaluate results. ON: Evaluate the results.

get_mscalar() → bool

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:MSCalar
value: bool = driver.configure.wlanMeas.multiEval.result.get_mscalar()
```

Enables or disables the evaluation of modulation scalar results.

return
modenable: OFF: Do not evaluate results. ON: Evaluate the results.

get_power_vs_time() → bool

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:PVTime
value: bool = driver.configure.wlanMeas.multiEval.result.get_power_vs_time()
```

Enables or disables the evaluation of power vs time results.

return
power_vs_time_enable: OFF: Do not evaluate results. ON: Evaluate the results.

get_spectr_flatness() → bool

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:SFLatness
value: bool = driver.configure.wlanMeas.multiEval.result.get_spectr_flatness()
```

Enables or disables the evaluation of spectrum flatness results.

return
spec_flatness: OFF: Do not evaluate results. ON: Evaluate the results.

get_ts_mask() → bool

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:TSMask
value: bool = driver.configure.wlanMeas.multiEval.result.get_ts_mask()
```

Enables or disables the evaluation of transmit spectrum mask results.

return
spec_enable: OFF: Do not evaluate results. ON: Evaluate the results.

get_ut_error() → bool

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:UTERror
value: bool = driver.configure.wlanMeas.multiEval.result.get_ut_error()
```

Enables or disables the evaluation of unused tone error results.

return
ute_enable: OFF: Do not evaluate results. ON: Evaluate the results.

set_all(*value: AllStruct*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult[:ALL]
structure = driver.configure.wlanMeas.multiEval.result.AllStruct()
structure.Mod_Scalar: bool = False
structure.Power_Vs_Time: bool = False
structure.Evm_Vs_Chip: bool = False
structure.Evm_Vs_Sym: bool = False
structure.Evm_Vs_Carr: bool = False
structure.Iq_Const: bool = False
structure.Spec_Flatness: bool = False
structure.Tran_Spec_Mask: bool = False
structure.Unused_Tone_Err: bool = False
driver.configure.wlanMeas.multiEval.result.set_all(value = structure)
```

Enables or disables the evaluation of results in the multi-evaluation measurement. This command combines all other CONFigure:WLAN:MEAS<i>:MEValuation:RESult... commands.

param value
see the help for AllStruct structure arguments.

set_evm(*evm_enable: bool*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:EVM
driver.configure.wlanMeas.multiEval.result.set_evm(evm_enable = False)
```

Enables or disables the evaluation of EVM vs chip results.

param evm_enable
OFF: Do not evaluate results. ON: Evaluate the results.

set_evm_carrier(*evm_enable: bool*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:EVMCarrier
driver.configure.wlanMeas.multiEval.result.set_evm_carrier(evm_enable = False)
```

Enables or disables the evaluation of EVM vs carrier results.

param evm_enable
OFF: Do not evaluate results. ON: Evaluate the results.

set_evm_symbol(*evm_enable: bool*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:EVMSymbol
driver.configure.wlanMeas.multiEval.result.set_evm_symbol(evm_enable = False)
```

Enables or disables the evaluation of EVM vs symbol results.

param evm_enable
OFF: Do not evaluate results. ON: Evaluate the results.

set_iq_constant(*iq_enable: bool*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:IQConst
driver.configure.wlanMeas.multiEval.result.set_iq_constant(iq_enable = False)
```

Enables or disables the evaluation of I/Q constellation results.

param iq_enable
OFF: Do not evaluate results. ON: Evaluate the results.

set_mscalar(*modenable: bool*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:MSCalar
driver.configure.wlanMeas.multiEval.result.set_mscalar(modenable = False)
```

Enables or disables the evaluation of modulation scalar results.

param modenable
OFF: Do not evaluate results. ON: Evaluate the results.

set_power_vs_time(*power_vs_time_enable: bool*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:PVTime
driver.configure.wlanMeas.multiEval.result.set_power_vs_time(power_vs_time_
↪enable = False)
```

Enables or disables the evaluation of power vs time results.

param power_vs_time_enable
OFF: Do not evaluate results. ON: Evaluate the results.

set_spectr_flatness(*spec_flatness: bool*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:SFLatness
driver.configure.wlanMeas.multiEval.result.set_spectr_flatness(spec_flatness =␣
↪False)
```

Enables or disables the evaluation of spectrum flatness results.

param spec_flatness
OFF: Do not evaluate results. ON: Evaluate the results.

set_ts_mask(*spec_enable: bool*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:TSMask
driver.configure.wlanMeas.multiEval.result.set_ts_mask(spec_enable = False)
```

Enables or disables the evaluation of transmit spectrum mask results.

param spec_enable
OFF: Do not evaluate results. ON: Evaluate the results.

set_ut_error(*ute_enable: bool*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:UTERror
driver.configure.wlanMeas.multiEval.result.set_ut_error(ute_enable = False)
```

Enables or disables the evaluation of unused tone error results.

param ute_enable
OFF: Do not evaluate results. ON: Evaluate the results.

6.1.1.3.7 Scount

SCPI Commands :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:SCOunt:TSMask
CONFigure:WLAN:MEASurement<Instance>:MEValuation:SCOunt:PVTime
CONFigure:WLAN:MEASurement<Instance>:MEValuation:SCOunt:MODulation
```

class ScountCls

Scount commands group definition. 3 total commands, 0 Subgroups, 3 group commands

get_modulation() → int

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:SCOunt:MODulation
value: int = driver.configure.wlanMeas.multiEval.scount.get_modulation()
```

Specifies the statistic count of the measurement. The statistic count is equal to the number of measurement intervals per single shot.

return
statistic_count: Number of measurement intervals for modulation measurements

get_power_vs_time() → int

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:SCOunt:PVTime
value: int = driver.configure.wlanMeas.multiEval.scount.get_power_vs_time()
```

Specifies the statistic count of the measurement. The statistic count is equal to the number of measurement intervals per single shot.

return
statistic_count: Number of measurement intervals for the power vs time measurement

get_ts_mask() → int

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:SCOunt:TSMask
value: int = driver.configure.wlanMeas.multiEval.scount.get_ts_mask()
```

Specifies the statistic count of the measurement. The statistic count is equal to the number of measurement intervals per single shot.

return
statistic_count: Number of measurement intervals for the transmit spectrum mask measurement

set_modulation(*statistic_count: int*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:SCOunt:MODulation
driver.configure.wlanMeas.multiEval.scount.set_modulation(statistic_count = 1)
```

Specifies the statistic count of the measurement. The statistic count is equal to the number of measurement intervals per single shot.

param statistic_count
Number of measurement intervals for modulation measurements

set_power_vs_time(*statistic_count: int*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:SCOunt:PVTime
driver.configure.wlanMeas.multiEval.scount.set_power_vs_time(statistic_count =␣
↪1)
```

Specifies the statistic count of the measurement. The statistic count is equal to the number of measurement intervals per single shot.

param statistic_count
Number of measurement intervals for the power vs time measurement

set_ts_mask(*statistic_count: int*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:SCOunt:TSMask
driver.configure.wlanMeas.multiEval.scount.set_ts_mask(statistic_count = 1)
```

Specifies the statistic count of the measurement. The statistic count is equal to the number of measurement intervals per single shot.

param statistic_count
Number of measurement intervals for the transmit spectrum mask measurement

6.1.1.3.8 SpectrFlatness

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:SFLatness:DMODe
```

class SpectrFlatnessCls

SpectrFlatness commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_dmode() → DisplayMode

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:SFLatness:DMODe
value: enums.DisplayMode = driver.configure.wlanMeas.multiEval.spectrFlatness.
↪get_dmode()
```

No command help available

return
disp_mode: No help available

set_dmode(*disp_mode: DisplayMode*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:SFLatness:DMODe
driver.configure.wlanMeas.multiEval.spectrFlatness.set_dmode(disp_mode = enums.
↪DisplayMode.ABSolute)
```

No command help available

param disp_mode
No help available

6.1.1.3.9 TsMask

SCPI Commands :

```
CONFigure:WLAN:MEASurement<Instance>:MEValuation:TSMask:AFFTnum
CONFigure:WLAN:MEASurement<Instance>:MEValuation:TSMask:TROTime
CONFigure:WLAN:MEASurement<Instance>:MEValuation:TSMask:OBWPercent
CONFigure:WLAN:MEASurement<Instance>:MEValuation:TSMask:MSELection
CONFigure:WLAN:MEASurement<Instance>:MEValuation:TSMask:DMODe
```

class TsMaskCls

TsMask commands group definition. 5 total commands, 0 Subgroups, 5 group commands

get_afft_num() → int

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:TSMask:AFFTnum
value: int = driver.configure.wlanMeas.multiEval.tsMask.get_afft_num()
```

Specifies the number of FFT operations per burst.

return
aver_fft_num: No help available

get_dmode() → DisplayMode

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:TSMask:DMODe
value: enums.DisplayMode = driver.configure.wlanMeas.multiEval.tsMask.get_
↪dmode()
```

Selects the display mode of Transmit Spectrum Mask results to switch between relative and absolute result values (dB vs dBm) .

return
disp_mode: No help available

get_mselection() → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:TSMask:MSELection
value: float = driver.configure.wlanMeas.multiEval.tsMask.get_mselection()
```

Selects the spectrum limit mask to be applied to 802.11p signals.

return
mask_selection: IEEE: Relative spectral density limits, IEEE Std 802.11-2020 ETSI: Absolute emission limits, ETSI EN 302 571 V1.1.1 (2008-09)

get_obw_percent() → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:TSMask:OBWPercent
value: float or bool = driver.configure.wlanMeas.multiEval.tsMask.get_obw_
↪percent()
```

Enables/disables OBW measurement and sets the OBW percentage.

return
obw_power: (float or boolean) No help available

get_tro_time() → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:TSMask:TROTime
value: float = driver.configure.wlanMeas.multiEval.tsMask.get_tro_time()
```

Specifies the trigger offset between trigger event and FFT operation.

return
trigger_off_time: No help available

set_afft_num(*aver_fft_num: int*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:TSMask:AFFTnum
driver.configure.wlanMeas.multiEval.tsMask.set_afft_num(aver_fft_num = 1)
```

Specifies the number of FFT operations per burst.

param aver_fft_num
No help available

set_dmode(*disp_mode: DisplayMode*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:TSMask:DMODe
driver.configure.wlanMeas.multiEval.tsMask.set_dmode(disp_mode = enums.
↪DisplayMode.ABSolute)
```

Selects the display mode of Transmit Spectrum Mask results to switch between relative and absolute result values (dB vs dBm) .

param disp_mode
No help available

set_mselection(*mask_selection: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:TSMask:MSELection
driver.configure.wlanMeas.multiEval.tsMask.set_mselection(mask_selection = 1.0)
```

Selects the spectrum limit mask to be applied to 802.11p signals.

param mask_selection
IEEE: Relative spectral density limits, IEEE Std 802.11-2020 ETSI: Absolute emission limits, ETSI EN 302 571 V1.1.1 (2008-09)

set_obw_percent(*obw_power: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:TSMask:OBWPercent
driver.configure.wlanMeas.multiEval.tsMask.set_obw_percent(obw_power = 1.0)
```

Enables/disables OBW measurement and sets the OBW percentage.

param obw_power
(float or boolean) No help available

set_tro_time(*trigger_off_time: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:MEValuation:TSMask:TROTime
driver.configure.wlanMeas.multiEval.tsMask.set_tro_time(trigger_off_time = 1.0)
```

Specifies the trigger offset between trigger event and FFT operation.

param trigger_off_time
No help available

6.1.1.4 RfSettings

SCPI Commands :

```
CONFigure:WLAN:MEASurement<Instance>:RFSettings:SANTennas
CONFigure:WLAN:MEASurement<Instance>:RFSettings:MLOFfset
CONFigure:WLAN:MEASurement<Instance>:RFSettings:FOFFset
CONFigure:WLAN:MEASurement<Instance>:RFSettings:LRINterval
```

class RfSettingsCls

RfSettings commands group definition. 13 total commands, 6 Subgroups, 4 group commands

get_foffset() → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:RFSettings:FOFFset
value: float = driver.configure.wlanMeas.rfSettings.get_foffset()
```

No command help available

return
freq_offset: No help available

get_lr_interval() → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:RFSettings:LRINterval
value: float = driver.configure.wlanMeas.rfSettings.get_lr_interval()
```

Defines the measurement interval for level adjustment.

return
lvl_rang_interval: No help available

get_ml_offset() → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:RFSettings:MLOFfset
value: float = driver.configure.wlanMeas.rfSettings.get_ml_offset()
```

No command help available

return
ml_offset: No help available

get_santennas() → bool

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:RFSettings:SANTennas
value: bool = driver.configure.wlanMeas.rfSettings.get_santennas()
```

No command help available

return
sep_ant: No help available

set_foffset(*freq_offset: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:RFSettings:FOFFset
driver.configure.wlanMeas.rfSettings.set_foffset(freq_offset = 1.0)
```

No command help available

param freq_offset
No help available

set_lr_interval(*lvl_rang_interval: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:RFSettings:LRINterval
driver.configure.wlanMeas.rfSettings.set_lr_interval(lvl_rang_interval = 1.0)
```

Defines the measurement interval for level adjustment.

param lvl_rang_interval
No help available

set_ml_offset(*ml_offset: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:RFSettings:MLOFfset
driver.configure.wlanMeas.rfSettings.set_ml_offset(ml_offset = 1.0)
```

No command help available

param ml_offset
No help available

set_santennas(*sep_ant: bool*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:RFSettings:SANTennas
driver.configure.wlanMeas.rfSettings.set_santennas(sep_ant = False)
```

No command help available

param sep_ant
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.rfSettings.clone()
```

Subgroups

6.1.1.4.1 Antenna<Antenna>

RepCap Settings

```
# Range: Nr1 .. Nr8
rc = driver.configure.wlanMeas.rfSettings.antenna.repcap_antenna_get()
driver.configure.wlanMeas.rfSettings.antenna.repcap_antenna_set(repcap.Antenna.Nr1)
```

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:RFSettings:ANTenna<n>
```

class AntennaCls

Antenna commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Antenna, default value after init: Antenna.Nr1

class GetStruct

Response structure. Fields:

- Connector_Smimo: enums.ConnectorSwitch: No parameter help available
- Connector_Tmimo: enums.RxConnectorExt: No parameter help available
- Ext_Att: float: No parameter help available
- Enp: float: No parameter help available

get(*antenna=Antenna.Default*) → GetStruct

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:RFSettings:ANTenna<n>
value: GetStruct = driver.configure.wlanMeas.rfSettings.antenna.get(antenna =␣
↪repcap.Antenna.Default)
```

No command help available

param antenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Antenna')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*connector_all: str*, *ext_att: float = None*, *enp: float = None*, *antenna=Antenna.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:RFSettings:ANTenna<n>
driver.configure.wlanMeas.rfSettings.antenna.set(connector_all = rawAbc, ext_
↪att = 1.0, enp = 1.0, antenna = repcap.Antenna.Default)
```

No command help available

param connector_all
No help available

param ext_att
No help available

param enp
No help available

param antenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Antenna')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.rfSettings.antenna.clone()
```

6.1.1.4.2 Eattenuation<Connector>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.configure.wlanMeas.rfSettings.eattenuation.repcap_connector_get()
driver.configure.wlanMeas.rfSettings.eattenuation.repcap_connector_set(repcap.Connector.
↪Nr1)
```

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:RFSettings:EATTenuation<antenna>
```

class EattenuationCls

Eattenuation commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Connector, default value after init: Connector.Nr1

get(*connector=Connector.Default*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:RFSettings:EATTenuation<antenna>
value: float = driver.configure.wlanMeas.rfSettings.eattenuation.get(connector␣
↪= repcap.Connector.Default)
```

Defines an external attenuation (or gain, if the value is negative) , to be applied to RF input connectors for SISO and MIMO connections.

param connector
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Eattenuation')

return
ext_attenuation: No help available

set(*ext_attenuation: float*, *connector=Connector.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:RFSettings:EATTenuation<antenna>
driver.configure.wlanMeas.rfSettings.eattenuation.set(ext_attenuation = 1.0,␣
↪connector = repcap.Connector.Default)
```

Defines an external attenuation (or gain, if the value is negative) , to be applied to RF input connectors for SISO and MIMO connections.

param ext_attenuation
No help available

param connector
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Eattenuation')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.rfSettings.eattenuation.clone()
```

6.1.1.4.3 EnvelopePower<Connector>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.configure.wlanMeas.rfSettings.envelopePower.repcap_connector_get()
driver.configure.wlanMeas.rfSettings.envelopePower.repcap_connector_set(repcap.Connector.
↪Nr1)
```

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:RFSettings:ENPower<antenna>
```

class EnvelopePowerCls

EnvelopePower commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Connector, default value after init: Connector.Nr1

get(*connector=Connector.Default*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:RFSettings:ENPower<antenna>
value: float = driver.configure.wlanMeas.rfSettings.envelopePower.get(connector␣
↪= repcap.Connector.Default)
```

Sets the expected nominal power of the measured RF signal.

param connector
optional repeated capability selector. Default value: Nr1 (settable in the interface 'EnvelopePower')

return
enp: The range of the expected nominal power can be calculated as follows: Range (Expected Nominal Power) = Range (Input Power) + External Attenuation - User Margin The input power range is stated in the specifications document.

set(*enp: float*, *connector=Connector.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:RFSettings:ENPower<antenna>
driver.configure.wlanMeas.rfSettings.envelopePower.set(enp = 1.0, connector =␣
↪repcap.Connector.Default)
```

Sets the expected nominal power of the measured RF signal.

param enp
The range of the expected nominal power can be calculated as follows: Range (Expected Nominal Power) = Range (Input Power) + External Attenuation - User Margin The input power range is stated in the specifications document.

param connector
optional repeated capability selector. Default value: Nr1 (settable in the interface 'EnvelopePower')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.rfSettings.envelopePower.clone()
```

6.1.1.4.4 Frequency

SCPI Commands :

```
CONFigure:WLAN:MEASurement<Instance>:RFSettings:FREQuency:SCHannel
CONFigure:WLAN:MEASurement<Instance>:RFSettings:FREQuency:BAND
CONFigure:WLAN:MEASurement<Instance>:RFSettings:FREQuency
```

class FrequencyCls

Frequency commands group definition. 4 total commands, 1 Subgroups, 3 group commands

get_band() → FrequencyBand

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:RFSettings:FREQuency:BAND
value: enums.FrequencyBand = driver.configure.wlanMeas.rfSettings.frequency.get_
↪band()
```

Selects the frequency band.

return
freq_band: B24Ghz: 2.4 GHz band B4GHz: 4 GHz band B5GHz: 5 GHz band B6GHz: 6 GHz band

get_schannel() → SlopeType

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:RFSettings:FREQuency:SCHannel
value: enums.SlopeType = driver.configure.wlanMeas.rfSettings.frequency.get_
↪schannel()
```

Sets the position of the secondary channel relative to the primary channel for 40 MHz 802.11n signals.

return
second_channel: POSitive: Secondary channel right above the primary channel NEGative: Secondary channel right below the primary channel

get_value() → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:RFSettings:FREQuency
value: float = driver.configure.wlanMeas.rfSettings.frequency.get_value()
```

Configures the center frequency of the RF analyzer. Set it to the center frequency of the received 20-MHz, 40-MHz, 80-MHz, 160-MHz, or 320-MHz WLAN channel.

return
frequency: For the supported frequency range, see 'Frequency ranges'.

set_band(*freq_band: FrequencyBand*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:RFSettings:FREQuency:BAND
driver.configure.wlanMeas.rfSettings.frequency.set_band(freq_band = enums.
↪FrequencyBand.B24Ghz)
```

Selects the frequency band.

param freq_band
B24Ghz: 2.4 GHz band B4GHz: 4 GHz band B5GHz: 5 GHz band B6GHz: 6 GHz band

set_schannel(*second_channel: SlopeType*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:RFSettings:FREQuency:SCHannel
driver.configure.wlanMeas.rfSettings.frequency.set_schannel(second_channel =␣
↪enums.SlopeType.NEGative)
```

Sets the position of the secondary channel relative to the primary channel for 40 MHz 802.11n signals.

param second_channel
POSitive: Secondary channel right above the primary channel NEGative: Secondary channel right below the primary channel

set_value(*frequency: float*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:RFSettings:FREQuency
driver.configure.wlanMeas.rfSettings.frequency.set_value(frequency = 1.0)
```

Configures the center frequency of the RF analyzer. Set it to the center frequency of the received 20-MHz, 40-MHz, 80-MHz, 160-MHz, or 320-MHz WLAN channel.

param frequency
For the supported frequency range, see 'Frequency ranges'.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.rfSettings.frequency.clone()
```

Subgroups

6.1.1.4.4.1 Channels<Channels>

RepCap Settings

```
# Range: Nr1 .. Nr8
rc = driver.configure.wlanMeas.rfSettings.frequency.channels.repcap_channels_get()
driver.configure.wlanMeas.rfSettings.frequency.channels.repcap_channels_set(repcap.
↪Channels.Nr1)
```

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:RFSettings:FREQuency:CHANnels<Ch>
```

class ChannelsCls

Channels commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Channels, default value after init: Channels.Nr1

class GetStruct

Response structure. Fields:

- Channel_Other: int: No parameter help available
- Channels: List[int]: Comma-separated list of channel indices 1, 2, 4, or 8 values, see table below.

get(*channels=Channels.Default*) → GetStruct

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:RFSettings:FREQuency:CHANnels<Ch>
value: GetStruct = driver.configure.wlanMeas.rfSettings.frequency.channels.
↪get(channels = repcap.Channels.Default)
```

The command logic depends on the standard. This description applies to the standards 802.11ac, ax, and be. For other standards, see method RsCMPX_WlanMeas.Configure.WlanMeas.RfSettings.Frequency.Channels.set. A setting command sets the channel number <Ch> to the channel index <Channel>. The other 20-MHz channels of the bandwidth are configured automatically, resulting in a sequence of channel indices with the increment 4, see examples. A query returns the channel indices of all 20-MHz channels as a comma-separated list.

INTRO_CMD_HELP: Before using this command, configure the standard, the bandwidth and the band, see:

- method RsCMPX_WlanMeas.Configure.WlanMeas.Isignal.standard
- method RsCMPX_WlanMeas.Configure.WlanMeas.Isignal.bandwidth
- method RsCMPX_WlanMeas.Configure.WlanMeas.RfSettings.Frequency.band

param channels
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Channels')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*channel: float*, *channels=Channels.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:RFSettings:FREQuency:CHANnels<Ch>
driver.configure.wlanMeas.rfSettings.frequency.channels.set(channel = 1.0,␣
↪channels = repcap.Channels.Default)
```

The command logic depends on the standard. This description applies to the standards 802.11ac, ax, and be. For other standards, see method RsCMPX_WlanMeas.Configure.WlanMeas.RfSettings.Frequency.Channels.set. A setting command sets the channel number <Ch> to the channel index <Channel>. The other 20-MHz channels of the bandwidth are configured automatically, resulting in a sequence of channel indices with the increment 4, see examples. A query returns the channel indices of all 20-MHz channels as a comma-separated list.

INTRO_CMD_HELP: Before using this command, configure the standard, the bandwidth and the band, see:

- method RsCMPX_WlanMeas.Configure.WlanMeas.Isignal.standard
- method RsCMPX_WlanMeas.Configure.WlanMeas.Isignal.bandwidth
- method RsCMPX_WlanMeas.Configure.WlanMeas.RfSettings.Frequency.band

param channel
Channel index for the 20-MHz channel number For a valid configuration, all 20-MHz channels must fit into the band. So the effective ranges depend on Ch, see table below.

param channels
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Channels')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.rfSettings.frequency.channels.clone()
```

6.1.1.4.5 LrStart

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:RFSettings:LRSTart
```

class LrStartCls

LrStart commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:RFSettings:LRSTart
driver.configure.wlanMeas.rfSettings.lrStart.set()
```

Starts level adjustment.

set_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:RFSettings:LRSTart
driver.configure.wlanMeas.rfSettings.lrStart.set_with_opc()
```

Starts level adjustment.

Same as set, but waits for the operation to complete before continuing further. Use the RsCMPX_WlanMeas.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

6.1.1.4.6 Umargin<Connector>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.configure.wlanMeas.rfSettings.umargin.repcap_connector_get()
driver.configure.wlanMeas.rfSettings.umargin.repcap_connector_set(repcap.Connector.Nr1)
```

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:RFSettings:UMARgin<antenna>
```

class UmarginCls

Umargin commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Connector, default value after init: Connector.Nr1

get(*connector=Connector.Default*) → float

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:RFSettings:UMARgin<antenna>
value: float = driver.configure.wlanMeas.rfSettings.umargin.get(connector =␣
↪repcap.Connector.Default)
```

Sets the margin that the measurement adds to the expected nominal power to determine the reference power. The reference power minus the external input attenuation must be within the power range of the selected input connector. Refer to the specifications document.

param connector
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Umargin')

return
user_margin: No help available

set(*user_margin: float*, *connector=Connector.Default*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:RFSettings:UMARgin<antenna>
driver.configure.wlanMeas.rfSettings.umargin.set(user_margin = 1.0, connector =␣
↪repcap.Connector.Default)
```

Sets the margin that the measurement adds to the expected nominal power to determine the reference power. The reference power minus the external input attenuation must be within the power range of the selected input connector. Refer to the specifications document.

param user_margin
No help available

param connector
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Umargin')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.rfSettings.umargin.clone()
```

6.1.1.5 Smimo

SCPI Command :

```
CONFigure:WLAN:MEASurement<instance>:SMIMo:CTUPle
```

class SmimoCls

Smimo commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_ctuple() → ConnectorTuple

```
# SCPI: CONFigure:WLAN:MEASurement<instance>:SMIMo:CTUPle
value: enums.ConnectorTuple = driver.configure.wlanMeas.smimo.get_ctuple()
```

No command help available

return
con_tuple: No help available

set_ctuple(*con_tuple: ConnectorTuple*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<instance>:SMIMo:CTUPle
driver.configure.wlanMeas.smimo.set_ctuple(con_tuple = enums.ConnectorTuple.
↪CT12)
```

No command help available

param con_tuple
No help available

6.1.1.6 Tmode

SCPI Command :

```
CONFigure:WLAN:MEASurement<Instance>:TMODe:NOANtennas
```

class TmodeCls

Tmode commands group definition. 3 total commands, 1 Subgroups, 1 group commands

get_no_antennas() → int

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:TMODe:NOANtennas
value: int = driver.configure.wlanMeas.tmode.get_no_antennas()
```

No command help available

return
no_of_antennas: No help available

set_no_antennas(*no_of_antennas: int*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<Instance>:TMODe:NOANtennas
driver.configure.wlanMeas.tmode.set_no_antennas(no_of_antennas = 1)
```

No command help available

param no_of_antennas
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlanMeas.tmode.clone()
```

Subgroups

6.1.1.6.1 File

SCPI Commands :

```
CONFigure:WLAN:MEASurement<instance>:TMODe:FILE:SAVE
CONFigure:WLAN:MEASurement<instance>:TMODe:FILE:DATE
```

class FileCls

File commands group definition. 2 total commands, 0 Subgroups, 2 group commands

get_date() → str

```
# SCPI: CONFigure:WLAN:MEASurement<instance>:TMODe:FILE:DATE
value: str = driver.configure.wlanMeas.tmode.file.get_date()
```

No command help available

return
file_date: No help available

get_save() → str

```
# SCPI: CONFigure:WLAN:MEASurement<instance>:TMODe:FILE:SAVE
value: str = driver.configure.wlanMeas.tmode.file.get_save()
```

No command help available

return
filename: No help available

set_save(*filename: str*) → None

```
# SCPI: CONFigure:WLAN:MEASurement<instance>:TMODe:FILE:SAVE
driver.configure.wlanMeas.tmode.file.set_save(filename = 'abc')
```

No command help available

param filename
No help available

6.2 Route

class RouteCls

Route commands group definition. 9 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.route.clone()
```

Subgroups

6.2.1 WlanMeas

SCPI Commands :

```
ROUTe:WLAN:MEASurement<Instance>:SMIMo
ROUTe:WLAN:MEASurement<Instance>
```

class WlanMeasCls

WlanMeas commands group definition. 9 total commands, 2 Subgroups, 2 group commands

class SmimoStruct

Structure for reading output parameters. Fields:

- Gui_Scenario: enums.GuiScenario: No parameter help available
- Controller: str: No parameter help available
- Rx_Connector_1: enums.RxConnectorExt: No parameter help available
- Rx_Converter_1: enums.RxConverter: No parameter help available
- Rx_Connector_2: enums.RxConnectorExt: No parameter help available
- Rx_Converter_2: enums.RxConverter: No parameter help available
- Rx_Connector_3: enums.RxConnectorExt: No parameter help available
- Rx_Converter_3: enums.RxConverter: No parameter help available
- Rx_Connector_4: enums.RxConnectorExt: No parameter help available
- Rx_Converter_4: enums.RxConverter: No parameter help available

class ValueStruct

Structure for reading output parameters. Fields:

- Scenario: enums.MimoScenario: No parameter help available
- Controller: str: No parameter help available
- Rx_Connector_1: enums.RxConnectorExt: No parameter help available

- Rx_Converter_1: enums.RxConverter: No parameter help available

get_smimo() → SmimoStruct

```
# SCPI: ROUTe:WLAN:MEASurement<Instance>:SMIMo
value: SmimoStruct = driver.route.wlanMeas.get_smimo()
```

No command help available

return
structure: for return value, see the help for SmimoStruct structure arguments.

get_value() → ValueStruct

```
# SCPI: ROUTe:WLAN:MEASurement<Instance>
value: ValueStruct = driver.route.wlanMeas.get_value()
```

No command help available

return
structure: for return value, see the help for ValueStruct structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.route.wlanMeas.clone()
```

Subgroups

6.2.1.1 Catalog

SCPI Command :

```
ROUTe:WLAN:MEASurement<Instance>:CATalog:SCENario
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_scenario() → List[GuiScenario]

```
# SCPI: ROUTe:WLAN:MEASurement<Instance>:CATalog:SCENario
value: List[enums.GuiScenario] = driver.route.wlanMeas.catalog.get_scenario()
```

No command help available

return
valid_gui_scenarios: No help available

6.2.1.2 Scenario

SCPI Commands :

```
ROUTe:WLAN:MEASurement<Instance>:SCENario:CSPath
ROUTe:WLAN:MEASurement<Instance>:SCENario
```

class ScenarioCls

Scenario commands group definition. 6 total commands, 4 Subgroups, 2 group commands

get_cspath() → str

```
# SCPI: ROUTe:WLAN:MEASurement<Instance>:SCENario:CSPath
value: str = driver.route.wlanMeas.scenario.get_cspath()
```

No command help available

return
master: No help available

get_value() → GuiScenario

```
# SCPI: ROUTe:WLAN:MEASurement<Instance>:SCENario
value: enums.GuiScenario = driver.route.wlanMeas.scenario.get_value()
```

No command help available

return
gui_scenario: No help available

set_cspath(*master: str*) → None

```
# SCPI: ROUTe:WLAN:MEASurement<Instance>:SCENario:CSPath
driver.route.wlanMeas.scenario.set_cspath(master = 'abc')
```

No command help available

param master
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.route.wlanMeas.scenario.clone()
```

Subgroups

6.2.1.2.1 Salone

SCPI Command :

```
ROUTe:WLAN:MEASurement<Instance>:SCENario:SALone
```

class SaloneCls

Salone commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class SaloneStruct

Response structure. Fields:

- Rx_Connector: enums.RxConnectorExt: No parameter help available
- Rx_Converter: enums.RxConverter: No parameter help available

get() → SaloneStruct

```
# SCPI: ROUTe:WLAN:MEASurement<Instance>:SCENario:SALone
value: SaloneStruct = driver.route.wlanMeas.scenario.salone.get()
```

No command help available

return
structure: for return value, see the help for SaloneStruct structure arguments.

set(*rx_connector: RxConnectorExt*, *rx_converter: RxConverter*) → None

```
# SCPI: ROUTe:WLAN:MEASurement<Instance>:SCENario:SALone
driver.route.wlanMeas.scenario.salone.set(rx_connector = enums.RxConnectorExt.
↪I11I, rx_converter = enums.RxConverter.IRX1)
```

No command help available

param rx_connector
No help available

param rx_converter
No help available

6.2.1.2.2 Smi<Smi>

RepCap Settings

```
# Range: Nr4 .. Nr4
rc = driver.route.wlanMeas.scenario.smi.repcap_smi_get()
driver.route.wlanMeas.scenario.smi.repcap_smi_set(repcap.Smi.Nr4)
```

SCPI Command :

```
ROUTe:WLAN:MEASurement<Instance>:SCENario:SMI<nr>
```

class SmiCls

Smi commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Smi, default value after init: Smi.Nr4

class SmiStruct

Structure for setting input parameters. Fields:

- Rx_Connector_1: enums.RxConnectorExt: No parameter help available
- Rx_Converter_1: enums.RxConverter: No parameter help available
- Rx_Connector_2: enums.RxConnectorExt: No parameter help available
- Rx_Converter_2: enums.RxConverter: No parameter help available
- Rx_Connector_3: enums.RxConnector: No parameter help available
- Rx_Converter_3: enums.RxTxConverter: No parameter help available
- Rx_Connector_4: enums.RxConnectorExt: No parameter help available
- Rx_Converter_4: enums.RxConverter: No parameter help available

get(*smi=Smi.Default*) → SmiStruct

```
# SCPI: ROUTe:WLAN:MEASurement<Instance>:SCENario:SMI<nr>
value: SmiStruct = driver.route.wlanMeas.scenario.smi.get(smi = repcap.Smi.
↪Default)
```

No command help available

param smi
optional repeated capability selector. Default value: Nr4 (settable in the interface 'Smi')

return
structure: for return value, see the help for SmiStruct structure arguments.

set(*structure: SmiStruct*, *smi=Smi.Default*) → None

```
# SCPI: ROUTe:WLAN:MEASurement<Instance>:SCENario:SMI<nr>
structure = driver.route.wlanMeas.scenario.smi.SmiStruct()
structure.Rx_Connector_1: enums.RxConnectorExt = enums.RxConnectorExt.I11I
structure.Rx_Converter_1: enums.RxConverter = enums.RxConverter.IRX1
structure.Rx_Connector_2: enums.RxConnectorExt = enums.RxConnectorExt.I11I
structure.Rx_Converter_2: enums.RxConverter = enums.RxConverter.IRX1
structure.Rx_Connector_3: enums.RxConnector = enums.RxConnector.I11I
structure.Rx_Converter_3: enums.RxTxConverter = enums.RxTxConverter.IRX1
structure.Rx_Connector_4: enums.RxConnectorExt = enums.RxConnectorExt.I11I
structure.Rx_Converter_4: enums.RxConverter = enums.RxConverter.IRX1
driver.route.wlanMeas.scenario.smi.set(structure, smi = repcap.Smi.Default)
```

No command help available

param structure
for set value, see the help for SmiStruct structure arguments.

param smi
optional repeated capability selector. Default value: Nr4 (settable in the interface 'Smi')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.route.wlanMeas.scenario.smi.clone()
```

6.2.1.2.3 Smimo<SMimoPath>

RepCap Settings

```
# Range: Count2 .. Count8
rc = driver.route.wlanMeas.scenario.smimo.repcap_sMimoPath_get()
driver.route.wlanMeas.scenario.smimo.repcap_sMimoPath_set(repcap.SMimoPath.Count2)
```

SCPI Command :

```
ROUTe:WLAN:MEASurement<Instance>:SCENario:SMIMo<PathCount>
```

class SmimoCls

Smimo commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: SMimoPath, default value after init: SMimoPath.Count2

class GetStruct

Response structure. Fields:

- Gui_Scenario: enums.GuiScenario: No parameter help available
- Con_Tuple: enums.ConnectorTuple: No parameter help available

get(*sMimoPath=SMimoPath.Default*) → GetStruct

```
# SCPI: ROUTe:WLAN:MEASurement<Instance>:SCENario:SMIMo<PathCount>
value: GetStruct = driver.route.wlanMeas.scenario.smimo.get(sMimoPath = repcap.
↪SMimoPath.Default)
```

No command help available

param sMimoPath
optional repeated capability selector. Default value: Count2 (settable in the interface 'Smimo')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*con_tuple: ConnectorTuple = None*, *sMimoPath=SMimoPath.Default*) → None

```
# SCPI: ROUTe:WLAN:MEASurement<Instance>:SCENario:SMIMo<PathCount>
driver.route.wlanMeas.scenario.smimo.set(con_tuple = enums.ConnectorTuple.CT12,␣
↪sMimoPath = repcap.SMimoPath.Default)
```

No command help available

param con_tuple
No help available

param sMimoPath
optional repeated capability selector. Default value: Count2 (settable in the interface 'Smimo')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.route.wlanMeas.scenario.smimo.clone()
```

6.2.1.2.4 Tmimo<TrueMimoPath>

RepCap Settings

```
# Range: Count1 .. Count4
rc = driver.route.wlanMeas.scenario.tmimo.repcap_trueMimoPath_get()
driver.route.wlanMeas.scenario.tmimo.repcap_trueMimoPath_set(repcap.TrueMimoPath.Count1)
```

SCPI Command :

```
ROUTe:WLAN:MEASurement<Instance>:SCENario:TMIMo<PathCount>
```

class TmimoCls

Tmimo commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: TrueMimoPath, default value after init: TrueMimoPath.Count1

get(*trueMimoPath=TrueMimoPath.Default*) → GuiScenario

```
# SCPI: ROUTe:WLAN:MEASurement<Instance>:SCENario:TMIMo<PathCount>
value: enums.GuiScenario = driver.route.wlanMeas.scenario.tmimo.
↪get(trueMimoPath = repcap.TrueMimoPath.Default)
```

No command help available

param trueMimoPath
optional repeated capability selector. Default value: Count1 (settable in the interface 'Tmimo')

return
gui_scenario: No help available

set(*trueMimoPath=TrueMimoPath.Default*) → None

```
# SCPI: ROUTe:WLAN:MEASurement<Instance>:SCENario:TMIMo<PathCount>
driver.route.wlanMeas.scenario.tmimo.set(trueMimoPath = repcap.TrueMimoPath.
↪Default)
```

No command help available

param trueMimoPath
optional repeated capability selector. Default value: Count1 (settable in the interface 'Tmimo')

set_with_opc(*trueMimoPath=TrueMimoPath.Default*, *opc_timeout_ms: int = -1*) → None

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.route.wlanMeas.scenario.tmimo.clone()
```

6.3 Trigger

class TriggerCls

Trigger commands group definition. 7 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.clone()
```

Subgroups

6.3.1 WlanMeas

class WlanMeasCls

WlanMeas commands group definition. 7 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.wlanMeas.clone()
```

Subgroups

6.3.1.1 MultiEval

SCPI Commands :

```
TRIGger:WLAN:MEASurement<Instance>:MEValuation:SOURce
TRIGger:WLAN:MEASurement<Instance>:MEValuation:MGAP
TRIGger:WLAN:MEASurement<Instance>:MEValuation:THReshold
TRIGger:WLAN:MEASurement<Instance>:MEValuation:SLOPe
TRIGger:WLAN:MEASurement<Instance>:MEValuation:TOUT
TRIGger:WLAN:MEASurement<Instance>:MEValuation:OFFSet
```

class MultiEvalCls

MultiEval commands group definition. 7 total commands, 1 Subgroups, 6 group commands

get_mgap() → float

```
# SCPI: TRIGger:WLAN:MEASurement<Instance>:MEValuation:MGAP
value: float = driver.trigger.wlanMeas.multiEval.get_mgap()
```

Sets a minimum time during which the IF signal must be below the trigger threshold before the trigger is armed so that an IF power trigger event can be generated.

return
min_trig_gap: No help available

get_offset() → float

```
# SCPI: TRIGger:WLAN:MEASurement<Instance>:MEValuation:OFFSet
value: float = driver.trigger.wlanMeas.multiEval.get_offset()
```

Sets the offset to the trigger event. The offset is useful if the trigger event and the burst are not synchronous. Triggering a measurement at another time can yield a synchronization error.

return
trig_offset: No help available

get_slope() → TriggerSlope

```
# SCPI: TRIGger:WLAN:MEASurement<Instance>:MEValuation:SLOPe
value: enums.TriggerSlope = driver.trigger.wlanMeas.multiEval.get_slope()
```

Qualifies whether the trigger event is generated at the rising or at the falling edge of the trigger pulse (valid for external and power trigger sources) .

return
trig_slope: REDGe: Rising edge FEDGe: Falling edge

get_source() → str

```
# SCPI: TRIGger:WLAN:MEASurement<Instance>:MEValuation:SOURce
value: str = driver.trigger.wlanMeas.multiEval.get_source()
```

Selects the source of the trigger events. Some values are always available. They are listed below. Depending on the installed options, additional values are available. You can query a list of all supported values via TRIGger:... :CATalog:SOURce?.

return
trig_source: 'IF Power': Power trigger (received RF power)

get_threshold() → float

```
# SCPI: TRIGger:WLAN:MEASurement<Instance>:MEValuation:THReshold
value: float or bool = driver.trigger.wlanMeas.multiEval.get_threshold()
```

Defines the trigger threshold for power trigger sources.

return
trig_threshold: (float or boolean) No help available

get_timeout() → float

```
# SCPI: TRIGger:WLAN:MEASurement<Instance>:MEValuation:TOUT
value: float or bool = driver.trigger.wlanMeas.multiEval.get_timeout()
```

Selects the maximum time that the measurement waits for a trigger event before it stops in remote control mode or indicates a trigger timeout in manual operation mode.

return
trig_time_out: (float or boolean) No help available

set_mgap(*min_trig_gap: float*) → None

```
# SCPI: TRIGger:WLAN:MEASurement<Instance>:MEValuation:MGAP
driver.trigger.wlanMeas.multiEval.set_mgap(min_trig_gap = 1.0)
```

Sets a minimum time during which the IF signal must be below the trigger threshold before the trigger is armed so that an IF power trigger event can be generated.

param min_trig_gap
No help available

set_offset(*trig_offset: float*) → None

```
# SCPI: TRIGger:WLAN:MEASurement<Instance>:MEValuation:OFFSet
driver.trigger.wlanMeas.multiEval.set_offset(trig_offset = 1.0)
```

Sets the offset to the trigger event. The offset is useful if the trigger event and the burst are not synchronous. Triggering a measurement at another time can yield a synchronization error.

param trig_offset
No help available

set_slope(*trig_slope: TriggerSlope*) → None

```
# SCPI: TRIGger:WLAN:MEASurement<Instance>:MEValuation:SLOPe
driver.trigger.wlanMeas.multiEval.set_slope(trig_slope = enums.TriggerSlope.
↪FEDGe)
```

Qualifies whether the trigger event is generated at the rising or at the falling edge of the trigger pulse (valid for external and power trigger sources) .

param trig_slope
REDGe: Rising edge FEDGe: Falling edge

set_source(*trig_source: str*) → None

```
# SCPI: TRIGger:WLAN:MEASurement<Instance>:MEValuation:SOURce
driver.trigger.wlanMeas.multiEval.set_source(trig_source = 'abc')
```

Selects the source of the trigger events. Some values are always available. They are listed below. Depending on the installed options, additional values are available. You can query a list of all supported values via TRIGger:... :CATalog:SOURce?.

param trig_source
'IF Power': Power trigger (received RF power)

set_threshold(*trig_threshold: float*) → None

```
# SCPI: TRIGger:WLAN:MEASurement<Instance>:MEValuation:THReshold
driver.trigger.wlanMeas.multiEval.set_threshold(trig_threshold = 1.0)
```

Defines the trigger threshold for power trigger sources.

param trig_threshold
(float or boolean) No help available

set_timeout(*trig_time_out: float*) → None

```
# SCPI: TRIGger:WLAN:MEASurement<Instance>:MEValuation:TOUT
driver.trigger.wlanMeas.multiEval.set_timeout(trig_time_out = 1.0)
```

Selects the maximum time that the measurement waits for a trigger event before it stops in remote control mode or indicates a trigger timeout in manual operation mode.

param trig_time_out
(float or boolean) No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.wlanMeas.multiEval.clone()
```

Subgroups

6.3.1.1.1 Catalog

class CatalogCls

Catalog commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.wlanMeas.multiEval.catalog.clone()
```

Subgroups

6.3.1.1.1.1 Source

SCPI Command :

```
TRIGger:WLAN:MEASurement<Instance>:MEValuation:CATalog:SOURce
```

class SourceCls

Source commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*full_list: bool = None*) → List[str]

```
# SCPI: TRIGger:WLAN:MEASurement<Instance>:MEValuation:CATalog:SOURce
value: List[str] = driver.trigger.wlanMeas.multiEval.catalog.source.get(full_
↪list = False)
```

Lists all trigger source values that can be set using method RsCMPX_WlanMeas.Trigger.WlanMeas.MultiEval.source.

param full_list
No help available

return
trig_source: Comma-separated list of all supported values. Each value is represented as a string.

6.4 WlanMeas

class WlanMeasCls

WlanMeas commands group definition. 1016 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.clone()
```

Subgroups

6.4.1 MultiEval

SCPI Commands :

```
STOP:WLAN:MEASurement<Instance>:MEValuation
ABORt:WLAN:MEASurement<Instance>:MEValuation
INITiate:WLAN:MEASurement<Instance>:MEValuation
```

class MultiEvalCls

MultiEval commands group definition. 1011 total commands, 11 Subgroups, 3 group commands

abort(*opc_timeout_ms: int = -1*) → None

```
# SCPI: ABORt:WLAN:MEASurement<Instance>:MEValuation
driver.wlanMeas.multiEval.abort()


    INTRO_CMD_HELP: Starts, stops or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the RUN state.
    - STOP... halts the measurement immediately. The measurement enters the RDY␣
```

(continues on next page)

(continued from previous page)

```
↪state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the␣
↪OFF state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

initiate(*opc_timeout_ms: int = -1*) → None

```
# SCPI: INITiate:WLAN:MEASurement<Instance>:MEValuation
driver.wlanMeas.multiEval.initiate()


    INTRO_CMD_HELP: Starts, stops or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the RUN state.
    - STOP... halts the measurement immediately. The measurement enters the RDY␣
↪state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the␣
↪OFF state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

stop(*opc_timeout_ms: int = -1*) → None

```
# SCPI: STOP:WLAN:MEASurement<Instance>:MEValuation
driver.wlanMeas.multiEval.stop()


    INTRO_CMD_HELP: Starts, stops or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the RUN state.
    - STOP... halts the measurement immediately. The measurement enters the RDY␣
↪state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the␣
↪OFF state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.clone()
```

Subgroups

6.4.1.1 ListPy

class ListPyCls

ListPy commands group definition. 132 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.listPy.clone()
```

Subgroups

6.4.1.1.1 Modulation

class ModulationCls

Modulation commands group definition. 112 total commands, 17 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.listPy.modulation.clone()
```

Subgroups

6.4.1.1.1.1 Bpower

class BpowerCls

Bpower commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.listPy.modulation.bpower.clone()
```

Subgroups

6.4.1.1.1.2 Average

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:BPOWer:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:BPOWer:AVERage
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.bpower.average.
↪fetch()
```

Return the burst power results for OFDM/OFDMA signals in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
burst_power: No help available

6.4.1.1.1.3 Current

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:BPOWer:CURRent
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:BPOWer:CURRent
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.bpower.current.
↪fetch()
```

Return the burst power results for OFDM/OFDMA signals in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
burst_power: No help available

6.4.1.1.1.4 Maximum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:BPOWer:MAXimum
```

class MaximumCls

Maximum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:BPOWer:MAXimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.bpower.maximum.
↪fetch()
```

Return the burst power results for OFDM/OFDMA signals in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
burst_power: No help available

6.4.1.1.1.5 Minimum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:BPOWer:MINimum
```

class MinimumCls

Minimum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:BPOWer:MINimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.bpower.minimum.
↪fetch()
```

Return the burst power results for OFDM/OFDMA signals in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
burst_power: No help available

6.4.1.1.1.6 StandardDev

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:BPOWer:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:BPOWer:SDEViation
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.bpower.
↪standardDev.fetch()
```

Return the burst power results for OFDM/OFDMA signals in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
burst_power: No help available

6.4.1.1.1.7 Cfactor

class CfactorCls

Cfactor commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.listPy.modulation.cfactor.clone()
```

Subgroups

6.4.1.1.1.8 Average

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:CFACtor:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:CFACtor:AVERage
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.cfactor.
↪average.fetch()
```

Return the crest factor results in list mode. The values in curly brackets {} are specified for each active segment: {... }seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure. WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
crest_factor: No help available

6.4.1.1.1.9 Current

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:CFACtor:CURRent
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:CFACtor:CURRent
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.cfactor.
↪current.fetch()
```

Return the crest factor results in list mode. The values in curly brackets {} are specified for each active segment: {... }seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure. WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
crest_factor: No help available

6.4.1.1.1.10 Maximum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:CFACtor:MAXimum
```

class MaximumCls

Maximum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:CFACtor:MAXimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.cfactor.
↪maximum.fetch()
```

Return the crest factor results in list mode. The values in curly brackets {} are specified for each active segment: {... }seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure. WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
crest_factor: No help available

6.4.1.1.1.11 Minimum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:CFACtor:MINimum
```

class MinimumCls

Minimum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:CFACtor:MINimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.cfactor.
↪minimum.fetch()
```

Return the crest factor results in list mode. The values in curly brackets {} are specified for each active segment: {... }seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure. WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
crest_factor: No help available

6.4.1.1.1.12 StandardDev

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:CFACtor:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:CFACtor:SDEViation
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.cfactor.
↪standardDev.fetch()
```

Return the crest factor results in list mode. The values in curly brackets {} are specified for each active segment: {... }seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure. WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
crest_factor: No help available

6.4.1.1.1.13 CfError

class CfErrorCls

CfError commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.listPy.modulation.cfError.clone()
```

Subgroups

6.4.1.1.1.14 Average

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:CFERror:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:CFERror:AVERage
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.cfError.
↪average.fetch()
```

Return the center frequency error results for OFDM/OFDMA signals in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
freq_error: No help available

6.4.1.1.1.15 Current

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:CFERror:CURRent
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:CFERror:CURRent
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.cfError.
↪current.fetch()
```

Return the center frequency error results for OFDM/OFDMA signals in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
freq_error: No help available

6.4.1.1.1.16 Maximum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:CFERror:MAXimum
```

class MaximumCls

Maximum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:CFERror:MAXimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.cfError.
↪maximum.fetch()
```

Return the center frequency error results for OFDM/OFDMA signals in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
freq_error: No help available

6.4.1.1.1.17 Minimum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:CFERror:MINimum
```

class MinimumCls

Minimum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:CFERror:MINimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.cfError.
↪minimum.fetch()
```

Return the center frequency error results for OFDM/OFDMA signals in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
freq_error: No help available

6.4.1.1.1.18 StandardDev

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:CFERror:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:CFERror:SDEViation
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.cfError.
↪standardDev.fetch()
```

Return the center frequency error results for OFDM/OFDMA signals in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
freq_error: No help available

6.4.1.1.1.19 DcPower

class DcPowerCls

DcPower commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.listPy.modulation.dcPower.clone()
```

Subgroups

6.4.1.1.1.20 Average

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DCPower:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DCPower:AVERage
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dcPower.
↪average.fetch()
```

Return the power results of DC subcarriers in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas. Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
dc_power: No help available

6.4.1.1.1.21 Current

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DCPower:CURRent
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DCPower:CURRent
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dcPower.
↪current.fetch()
```

Return the power results of DC subcarriers in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas. Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
dc_power: No help available

6.4.1.1.1.22 Maximum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DCPower:MAXimum
```

class MaximumCls

Maximum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DCPower:MAXimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dcPower.
↪maximum.fetch()
```

Return the power results of DC subcarriers in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas. Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
dc_power: No help available

6.4.1.1.1.23 Minimum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DCPower:MINimum
```

class MinimumCls

Minimum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DCPower:MINimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dcPower.
↪minimum.fetch()
```

Return the power results of DC subcarriers in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas. Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
dc_power: No help available

6.4.1.1.1.24 StandardDev

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DCPower:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DCPower:SDEViation
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dcPower.
↪standardDev.fetch()
```

Return the power results of DC subcarriers in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas. Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
dc_power: No help available

6.4.1.1.1.25 Dpower

class DpowerCls

Dpower commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.listPy.modulation.dpower.clone()
```

Subgroups

6.4.1.1.1.26 Average

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DPOWer:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DPOWer:AVERage
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dpower.average.
↪fetch()
```

Return the power results of the data portion of the burst in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
data_power: No help available

6.4.1.1.1.27 Current

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DPOWer:CURRent
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DPOWer:CURRent
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dpower.current.
↪fetch()
```

Return the power results of the data portion of the burst in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
data_power: No help available

6.4.1.1.1.28 Maximum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DPOWer:MAXimum
```

class MaximumCls

Maximum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DPOWer:MAXimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dpower.maximum.
↪fetch()
```

Return the power results of the data portion of the burst in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
data_power: No help available

6.4.1.1.1.29 Minimum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DPOWer:MINimum
```

class MinimumCls

Minimum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DPOWer:MINimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dpower.minimum.
↪fetch()
```

Return the power results of the data portion of the burst in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
data_power: No help available

6.4.1.1.1.30 StandardDev

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DPOWer:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DPOWer:SDEViation
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dpower.
↪standardDev.fetch()
```

Return the power results of the data portion of the burst in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
data_power: No help available

6.4.1.1.1.31 Dsss

class DsssCls

Dsss commands group definition. 40 total commands, 8 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.listPy.modulation.dsss.clone()
```

Subgroups

6.4.1.1.1.32 Bpower

class BpowerCls

Bpower commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.listPy.modulation.dsss.bpower.clone()
```

Subgroups

6.4.1.1.1.33 Average

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DSSS:BPOWer:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DSSS:BPOWer:AVERage
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dsss.bpower.
↪average.fetch()
```

Return the burst power results for DSSS signals in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
burst_power: No help available

6.4.1.1.1.34 Current

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DSSS:BPOWer:CURRent
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DSSS:BPOWer:CURRent
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dsss.bpower.
↪current.fetch()
```

Return the burst power results for DSSS signals in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
burst_power: No help available

6.4.1.1.1.35 Maximum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DSSS:BPOWer:MAXimum
```

class MaximumCls

Maximum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DSSS:BPOWer:MAXimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dsss.bpower.
↪maximum.fetch()
```

Return the burst power results for DSSS signals in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
burst_power: No help available

6.4.1.1.1.36 Minimum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DSSS:BPOWer:MINimum
```

class MinimumCls

Minimum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DSSS:BPOWer:MINimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dsss.bpower.
↪minimum.fetch()
```

Return the burst power results for DSSS signals in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
burst_power: No help available

6.4.1.1.1.37 StandardDev

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DSSS:BPOWer:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DSSS:BPOWer:SDEViation
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dsss.bpower.
↪standardDev.fetch()
```

Return the burst power results for DSSS signals in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
burst_power: No help available

6.4.1.1.1.38 CcError

class CcErrorCls

CcError commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.listPy.modulation.dsss.ccError.clone()
```

Subgroups

6.4.1.1.1.39 Average

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DSSS:CCERror:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DSSS:CCERror:AVERage
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dsss.ccError.
↪average.fetch()
```

Return the chip clock error results for DSSS signals in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
clock_error: No help available

6.4.1.1.1.40 Current

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DSSS:CCERror:CURRent
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DSSS:CCERror:CURRent
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dsss.ccError.
↪current.fetch()
```

Return the chip clock error results for DSSS signals in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
clock_error: No help available

6.4.1.1.1.41 Maximum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DSSS:CCERror:MAXimum
```

class MaximumCls

Maximum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DSSS:CCERror:MAXimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dsss.ccError.
↪maximum.fetch()
```

Return the chip clock error results for DSSS signals in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
clock_error: No help available

6.4.1.1.1.42 Minimum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DSSS:CCERror:MINimum
```

class MinimumCls

Minimum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DSSS:CCERror:MINimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dsss.ccError.
↪minimum.fetch()
```

Return the chip clock error results for DSSS signals in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
clock_error: No help available

6.4.1.1.1.43 StandardDev

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DSSS:CCERror:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DSSS:CCERror:SDEViation
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dsss.ccError.
↪standardDev.fetch()
```

Return the chip clock error results for DSSS signals in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
clock_error: No help available

6.4.1.1.1.44 CfError

class CfErrorCls

CfError commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.listPy.modulation.dsss.cfError.clone()
```

Subgroups

6.4.1.1.1.45 Average

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DSSS:CFERror:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DSSS:CFERror:AVERage
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dsss.cfError.
↪average.fetch()
```

Return the center frequency error results for DSSS signals in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
freq_error: No help available

6.4.1.1.1.46 Current

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DSSS:CFERror:CURRent
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DSSS:CFERror:CURRent
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dsss.cfError.
↪current.fetch()
```

Return the center frequency error results for DSSS signals in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
freq_error: No help available

6.4.1.1.1.47 Maximum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DSSS:CFERror:MAXimum
```

class MaximumCls

Maximum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DSSS:CFERror:MAXimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dsss.cfError.
↪maximum.fetch()
```

Return the center frequency error results for DSSS signals in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
freq_error: No help available

6.4.1.1.1.48 Minimum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DSSS:CFERror:MINimum
```

class MinimumCls

Minimum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DSSS:CFERror:MINimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dsss.cfError.
↪minimum.fetch()
```

Return the center frequency error results for DSSS signals in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
freq_error: No help available

6.4.1.1.1.49 StandardDev

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DSSS:CFERror:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DSSS:CFERror:SDEViation
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dsss.cfError.
↪standardDev.fetch()
```

Return the center frequency error results for DSSS signals in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
freq_error: No help available

6.4.1.1.1.50 EvmEms

class EvmEmsCls

EvmEms commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.listPy.modulation.dsss.evmEms.clone()
```

Subgroups

6.4.1.1.1.51 Average

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DSSS:EVMRms:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DSSS:EVMRms:AVERage
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dsss.evmEms.
↪average.fetch()
```

Return the current, average, minimum, maximum and standard deviation EVM results for DSSS signals in list mode. Commands for EVM peak and EVM RMS values are available. The values in curly brackets {} are specified for each active segment: {... }seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure. WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
evm_rms: No help available

6.4.1.1.1.52 Current

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DSSS:EVMRms:CURRent
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DSSS:EVMRms:CURRent
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dsss.evmEms.
↪current.fetch()
```

Return the current, average, minimum, maximum and standard deviation EVM results for DSSS signals in list mode. Commands for EVM peak and EVM RMS values are available. The values in curly brackets {} are specified for each active segment: {... }seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure. WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
evm_rms: No help available

6.4.1.1.1.53 Maximum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DSSS:EVMRms:MAXimum
```

class MaximumCls

Maximum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DSSS:EVMRms:MAXimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dsss.evmEms.
↪maximum.fetch()
```

Return the current, average, minimum, maximum and standard deviation EVM results for DSSS signals in list mode. Commands for EVM peak and EVM RMS values are available. The values in curly brackets {} are specified for each active segment: {... }seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure. WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
evm_rms: No help available

6.4.1.1.1.54 Minimum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DSSS:EVMRms:MINimum
```

class MinimumCls

Minimum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DSSS:EVMRms:MINimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dsss.evmEms.
↪minimum.fetch()
```

Return the current, average, minimum, maximum and standard deviation EVM results for DSSS signals in list mode. Commands for EVM peak and EVM RMS values are available. The values in curly brackets {} are specified for each active segment: {... }seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure. WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
evm_rms: No help available

6.4.1.1.1.55 StandardDev

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DSSS:EVMRms:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DSSS:EVMRms:SDEViation
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dsss.evmEms.
↪standardDev.fetch()
```

Return the current, average, minimum, maximum and standard deviation EVM results for DSSS signals in list mode. Commands for EVM peak and EVM RMS values are available. The values in curly brackets {} are specified for each active segment: {... }seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure. WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
evm_rms: No help available

6.4.1.1.1.56 EvmPeak

class EvmPeakCls

EvmPeak commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.listPy.modulation.dsss.evmPeak.clone()
```

Subgroups

6.4.1.1.1.57 Average

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DSSS:EVMPeak:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DSSS:EVMPeak:AVERage
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dsss.evmPeak.
↪average.fetch()
```

Return the current, average, minimum, maximum and standard deviation EVM results for DSSS signals in list mode. Commands for EVM peak and EVM RMS values are available. The values in curly brackets {} are specified for each active segment: {... }seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure. WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
evm_peak: No help available

6.4.1.1.1.58 Current

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DSSS:EVMPeak:CURRent
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DSSS:EVMPeak:CURRent
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dsss.evmPeak.
↪current.fetch()
```

Return the current, average, minimum, maximum and standard deviation EVM results for DSSS signals in list mode. Commands for EVM peak and EVM RMS values are available. The values in curly brackets

{} are specified for each active segment: {... }seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure. WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
evm_peak: No help available

6.4.1.1.1.59 Maximum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DSSS:EVMPeak:MAXimum
```

class MaximumCls

Maximum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DSSS:EVMPeak:MAXimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dsss.evmPeak.
↪maximum.fetch()
```

Return the current, average, minimum, maximum and standard deviation EVM results for DSSS signals in list mode. Commands for EVM peak and EVM RMS values are available. The values in curly brackets {} are specified for each active segment: {... }seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure. WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
evm_peak: No help available

6.4.1.1.1.60 Minimum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DSSS:EVMPeak:MINimum
```

class MinimumCls

Minimum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DSSS:EVMPeak:MINimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dsss.evmPeak.
↪minimum.fetch()
```

Return the current, average, minimum, maximum and standard deviation EVM results for DSSS signals in list mode. Commands for EVM peak and EVM RMS values are available. The values in curly brackets {} are specified for each active segment: {... }seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure. WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
evm_peak: No help available

6.4.1.1.1.61 StandardDev

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DSSS:EVMPeak:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DSSS:EVMPeak:SDEViation
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dsss.evmPeak.
↪standardDev.fetch()
```

Return the current, average, minimum, maximum and standard deviation EVM results for DSSS signals in list mode. Commands for EVM peak and EVM RMS values are available. The values in curly brackets {} are specified for each active segment: {... }seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure. WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
evm_peak: No help available

6.4.1.1.1.62 Gimbalance

class GimbalanceCls

Gimbalance commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.listPy.modulation.dsss.gimbalance.clone()
```

Subgroups

6.4.1.1.1.63 Average

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DSSS:GIMBalance:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DSSS:GIMBalance:AVERage
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dsss.
↪gimbalance.average.fetch()
```

Return the gain imbalance results for DSSS signals in list mode. The values in curly brackets { } are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
gain_imbalance: No help available

6.4.1.1.1.64 Current

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DSSS:GIMBalance:CURRent
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DSSS:GIMBalance:CURRent
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dsss.
↪gimbalance.current.fetch()
```

Return the gain imbalance results for DSSS signals in list mode. The values in curly brackets { } are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
gain_imbalance: No help available

6.4.1.1.1.65 Maximum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DSSS:GIMBalance:MAXimum
```

class MaximumCls

Maximum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DSSS:GIMBalance:MAXimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dsss.
↪gimbalance.maximum.fetch()
```

Return the gain imbalance results for DSSS signals in list mode. The values in curly brackets { } are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
gain_imbalance: No help available

6.4.1.1.1.66 Minimum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DSSS:GIMBalance:MINimum
```

class MinimumCls

Minimum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DSSS:GIMBalance:MINimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dsss.
↪gimbalance.minimum.fetch()
```

Return the gain imbalance results for DSSS signals in list mode. The values in curly brackets { } are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
gain_imbalance: No help available

6.4.1.1.1.67 StandardDev

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DSSS:GIMBalance:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DSSS:GIMBalance:SDEViation
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dsss.
↪gimbalance.standardDev.fetch()
```

Return the gain imbalance results for DSSS signals in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
gain_imbalance: No help available

6.4.1.1.1.68 IqOffset

class IqOffsetCls

IqOffset commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.listPy.modulation.dsss.iqOffset.clone()
```

Subgroups

6.4.1.1.1.69 Average

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DSSS:IQOFfset:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DSSS:IQOFfset:AVERage
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dsss.iqOffset.
↪average.fetch()
```

Return the I/Q offset results for DSSS signals in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
iq_offset: No help available

6.4.1.1.1.70 Current

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DSSS:IQOFfset:CURRent
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DSSS:IQOFfset:CURRent
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dsss.iqOffset.
↪current.fetch()
```

Return the I/Q offset results for DSSS signals in list mode. The values in curly brackets { } are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
iq_offset: No help available

6.4.1.1.1.71 Maximum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DSSS:IQOFfset:MAXimum
```

class MaximumCls

Maximum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DSSS:IQOFfset:MAXimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dsss.iqOffset.
↪maximum.fetch()
```

Return the I/Q offset results for DSSS signals in list mode. The values in curly brackets { } are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
iq_offset: No help available

6.4.1.1.1.72 Minimum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DSSS:IQOFfset:MINimum
```

class MinimumCls

Minimum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DSSS:IQOFfset:MINimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dsss.iqOffset.
↪minimum.fetch()
```

Return the I/Q offset results for DSSS signals in list mode. The values in curly brackets { } are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
iq_offset: No help available

6.4.1.1.1.73 StandardDev

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DSSS:IQOFfset:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DSSS:IQOFfset:SDEViation
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dsss.iqOffset.
↪standardDev.fetch()
```

Return the I/Q offset results for DSSS signals in list mode. The values in curly brackets { } are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
iq_offset: No help available

6.4.1.1.1.74 Qerror

class QerrorCls

Qerror commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.listPy.modulation.dsss.qerror.clone()
```

Subgroups

6.4.1.1.1.75 Average

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DSSS:QERRor:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DSSS:QERRor:AVERage
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dsss.qerror.
↪average.fetch()
```

Return the quadrature error results for DSSS in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas. Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
quad_error: No help available

6.4.1.1.1.76 Current

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DSSS:QERRor:CURRent
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DSSS:QERRor:CURRent
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dsss.qerror.
↪current.fetch()
```

Return the quadrature error results for DSSS in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas. Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
quad_error: No help available

6.4.1.1.1.77 Maximum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DSSS:QERRor:MAXimum
```

class MaximumCls

Maximum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DSSS:QERRor:MAXimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dsss.qerror.
↪maximum.fetch()
```

Return the quadrature error results for DSSS in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas. Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
quad_error: No help available

6.4.1.1.1.78 Minimum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DSSS:QERRor:MINimum
```

class MinimumCls

Minimum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DSSS:QERRor:MINimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dsss.qerror.
↪minimum.fetch()
```

Return the quadrature error results for DSSS in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas. Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
quad_error: No help available

6.4.1.1.1.79 StandardDev

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:DSSS:QERRor:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:DSSS:QERRor:SDEViation
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.dsss.qerror.
↪standardDev.fetch()
```

Return the quadrature error results for DSSS in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas. Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
quad_error: No help available

6.4.1.1.1.80 EvmAll

class EvmAllCls

EvmAll commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.listPy.modulation.evmAll.clone()
```

Subgroups

6.4.1.1.1.81 Average

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMall:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVMall:AVERage
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.evmAll.average.
↪fetch()
```

Return the EVM results for OFDM/OFDMA signals for all carriers in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
evm_all: No help available

6.4.1.1.1.82 Current

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMall:CURRent
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVMall:CURRent
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.evmAll.current.
↪fetch()
```

Return the EVM results for OFDM/OFDMA signals for all carriers in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
evm_all: No help available

6.4.1.1.1.83 Maximum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMall:MAXimum
```

class MaximumCls

Maximum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVMall:MAXimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.evmAll.maximum.
↪fetch()
```

Return the EVM results for OFDM/OFDMA signals for all carriers in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
evm_all: No help available

6.4.1.1.1.84 Minimum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMall:MINimum
```

class MinimumCls

Minimum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVMall:MINimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.evmAll.minimum.
↪fetch()
```

Return the EVM results for OFDM/OFDMA signals for all carriers in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
evm_all: No help available

6.4.1.1.1.85 StandardDev

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMall:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVMall:SDEViation
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.evmAll.
↪standardDev.fetch()
```

Return the EVM results for OFDM/OFDMA signals for all carriers in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
evm_all: No help available

6.4.1.1.1.86 EvmData

class EvmDataCls

EvmData commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.listPy.modulation.evmData.clone()
```

Subgroups

6.4.1.1.1.87 Average

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMData:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVMData:AVERage
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.evmData.
↪average.fetch()
```

Return the EVM results for OFDM/OFDMA signals for data carriers in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
evm_data: No help available

6.4.1.1.1.88 Current

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMData:CURRent
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVMData:CURRent
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.evmData.
↪current.fetch()
```

Return the EVM results for OFDM/OFDMA signals for data carriers in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
evm_data: No help available

6.4.1.1.1.89 Maximum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMData:MAXimum
```

class MaximumCls

Maximum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVMData:MAXimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.evmData.
↪maximum.fetch()
```

Return the EVM results for OFDM/OFDMA signals for data carriers in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
evm_data: No help available

6.4.1.1.1.90 Minimum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMData:MINimum
```

class MinimumCls

Minimum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVMData:MINimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.evmData.
↪minimum.fetch()
```

Return the EVM results for OFDM/OFDMA signals for data carriers in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
evm_data: No help available

6.4.1.1.1.91 StandardDev

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMData:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVMData:SDEViation
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.evmData.
↪standardDev.fetch()
```

Return the EVM results for OFDM/OFDMA signals for data carriers in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
evm_data: No help available

6.4.1.1.1.92 EvmPilot

class EvmPilotCls

EvmPilot commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.listPy.modulation.evmPilot.clone()
```

Subgroups

6.4.1.1.1.93 Average

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMPilot:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVMPilot:AVERage
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.evmPilot.
↪average.fetch()
```

Return the EVM results for OFDM/OFDMA signals for pilot carrier in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
evm_pilot: No help available

6.4.1.1.1.94 Current

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMPilot:CURRent
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVMPilot:CURRent
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.evmPilot.
↪current.fetch()
```

Return the EVM results for OFDM/OFDMA signals for pilot carrier in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
evm_pilot: No help available

6.4.1.1.1.95 Maximum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMPilot:MAXimum
```

class MaximumCls

Maximum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVMPilot:MAXimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.evmPilot.
↪maximum.fetch()
```

Return the EVM results for OFDM/OFDMA signals for pilot carrier in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
evm_pilot: No help available

6.4.1.1.1.96 Minimum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMPilot:MINimum
```

class MinimumCls

Minimum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVMPilot:MINimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.evmPilot.
↪minimum.fetch()
```

Return the EVM results for OFDM/OFDMA signals for pilot carrier in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
evm_pilot: No help available

6.4.1.1.1.97 StandardDev

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMPilot:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVMPilot:SDEViation
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.evmPilot.
↪standardDev.fetch()
```

Return the EVM results for OFDM/OFDMA signals for pilot carrier in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
evm_pilot: No help available

6.4.1.1.1.98 Gimbalance

class GimbalanceCls

Gimbalance commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.listPy.modulation.gimbalance.clone()
```

Subgroups

6.4.1.1.1.99 Average

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:GIMBalance:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:GIMBalance:AVERage
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.gimbalance.
↪average.fetch()
```

Return the gain imbalance results for OFDM/OFDMA signals in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
gain_imbalance: No help available

6.4.1.1.1.100 Current

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:GIMBalance:CURRent
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:GIMBalance:CURRent
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.gimbalance.
↪current.fetch()
```

Return the gain imbalance results for OFDM/OFDMA signals in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
gain_imbalance: No help available

6.4.1.1.1.101 Maximum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:GIMBalance:MAXimum
```

class MaximumCls

Maximum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:GIMBalance:MAXimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.gimbalance.
↪maximum.fetch()
```

Return the gain imbalance results for OFDM/OFDMA signals in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
gain_imbalance: No help available

6.4.1.1.1.102 Minimum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:GIMBalance:MINimum
```

class MinimumCls

Minimum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:GIMBalance:MINimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.gimbalance.
↪minimum.fetch()
```

Return the gain imbalance results for OFDM/OFDMA signals in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
gain_imbalance: No help available

6.4.1.1.1.103 StandardDev

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:GIMBalance:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:GIMBalance:SDEViation
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.gimbalance.
↪standardDev.fetch()
```

Return the gain imbalance results for OFDM/OFDMA signals in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
gain_imbalance: No help available

6.4.1.1.1.104 IqOffset

class IqOffsetCls

IqOffset commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.listPy.modulation.iqOffset.clone()
```

Subgroups

6.4.1.1.1.105 Average

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:IQOFfset:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:IQOFfset:AVERage
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.iqOffset.
↪average.fetch()
```

Return the I/Q offset results for OFDM/OFDMA signals in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
iq_offset: No help available

6.4.1.1.1.106 Current

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:IQOFfset:CURRent
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:IQOFfset:CURRent
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.iqOffset.
↪current.fetch()
```

Return the I/Q offset results for OFDM/OFDMA signals in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
iq_offset: No help available

6.4.1.1.1.107 Maximum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:IQOFfset:MAXimum
```

class MaximumCls

Maximum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:IQOFfset:MAXimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.iqOffset.
↪maximum.fetch()
```

Return the I/Q offset results for OFDM/OFDMA signals in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
iq_offset: No help available

6.4.1.1.1.108 Minimum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:IQOFfset:MINimum
```

class MinimumCls

Minimum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:IQOFfset:MINimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.iqOffset.
↪minimum.fetch()
```

Return the I/Q offset results for OFDM/OFDMA signals in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
iq_offset: No help available

6.4.1.1.1.109 StandardDev

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:IQOFfset:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:IQOFfset:SDEViation
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.iqOffset.
↪standardDev.fetch()
```

Return the I/Q offset results for OFDM/OFDMA signals in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
iq_offset: No help available

6.4.1.1.1.110 LtfPower

class LtfPowerCls

LtfPower commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.listPy.modulation.ltfPower.clone()
```

Subgroups

6.4.1.1.1.111 Average

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:LTFPower:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:LTFPower:AVERage
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.ltfPower.
↪average.fetch()
```

Return the power results of the LTF portion of the burst in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
ltf_power: No help available

6.4.1.1.1.112 Current

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:LTFPower:CURRent
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:LTFPower:CURRent
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.ltfPower.
↪current.fetch()
```

Return the power results of the LTF portion of the burst in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
ltf_power: No help available

6.4.1.1.1.113 Maximum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:LTFPower:MAXimum
```

class MaximumCls

Maximum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:LTFPower:MAXimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.ltfPower.
↪maximum.fetch()
```

Return the power results of the LTF portion of the burst in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
ltf_power: No help available

6.4.1.1.1.114 Minimum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:LTFPower:MINimum
```

class MinimumCls

Minimum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:LTFPower:MINimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.ltfPower.
↪minimum.fetch()
```

Return the power results of the LTF portion of the burst in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
ltf_power: No help available

6.4.1.1.1.115 StandardDev

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:LTFPower:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:LTFPower:SDEViation
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.ltfPower.
↪standardDev.fetch()
```

Return the power results of the LTF portion of the burst in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
ltf_power: No help available

6.4.1.1.1.116 Pbackoff

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:PBACkoff
```

class PbackoffCls

Pbackoff commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:PBACkoff
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.pbackoff.
↪fetch()
```

Return the power backoff results in list mode. The power backoff displays the minimum distance of signal power to the reference level over all segments since the start of the measurement. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
power_backoff: No help available

6.4.1.1.1.117 Ppower

class PpowerCls

Ppower commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.listPy.modulation.ppower.clone()
```

Subgroups

6.4.1.1.1.118 Average

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:PPOWer:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PPOWer:AVERage
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.ppower.average.
↪fetch()
```

Return the peak power results of the burst in list mode. The values in curly brackets { } are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas. Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
peak_power: No help available

6.4.1.1.1.119 Current

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:PPOWer:CURRent
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PPOWer:CURRent
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.ppower.current.
↪fetch()
```

Return the peak power results of the burst in list mode. The values in curly brackets { } are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas. Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
peak_power: No help available

6.4.1.1.1.120 Maximum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:PPOWer:MAXimum
```

class MaximumCls

Maximum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PPOWer:MAXimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.ppower.maximum.
↪fetch()
```

Return the peak power results of the burst in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas. Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
peak_power: No help available

6.4.1.1.1.121 Minimum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:PPOWer:MINimum
```

class MinimumCls

Minimum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PPOWer:MINimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.ppower.minimum.
↪fetch()
```

Return the peak power results of the burst in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas. Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
peak_power: No help available

6.4.1.1.1.122 StandardDev

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:PPOWer:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PPOWer:SDEViation
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.ppower.
↪standardDev.fetch()
```

Return the peak power results of the burst in list mode. The values in curly brackets { } are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas. Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
peak_power: No help available

6.4.1.1.1.123 Qerror

class QerrorCls

Qerror commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.listPy.modulation.qerror.clone()
```

Subgroups

6.4.1.1.1.124 Average

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:QERRor:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:QERRor:AVERage
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.qerror.average.
↪fetch()
```

Return the quadrature error results for OFDM/OFDMA in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
quad_error: No help available

6.4.1.1.1.125 Current

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:QERRor:CURRent
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:QERRor:CURRent
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.qerror.current.
↪fetch()
```

Return the quadrature error results for OFDM/OFDMA in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
quad_error: No help available

6.4.1.1.1.126 Maximum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:QERRor:MAXimum
```

class MaximumCls

Maximum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:QERRor:MAXimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.qerror.maximum.
↪fetch()
```

Return the quadrature error results for OFDM/OFDMA in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
quad_error: No help available

6.4.1.1.1.127 Minimum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:QERRor:MINimum
```

class MinimumCls

Minimum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:QERRor:MINimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.qerror.minimum.
↪fetch()
```

Return the quadrature error results for OFDM/OFDMA in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
quad_error: No help available

6.4.1.1.1.128 StandardDev

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:QERRor:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:QERRor:SDEViation
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.qerror.
↪standardDev.fetch()
```

Return the quadrature error results for OFDM/OFDMA in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
quad_error: No help available

6.4.1.1.1.129 ScError

class ScErrorCls

ScError commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.listPy.modulation.scError.clone()
```

Subgroups

6.4.1.1.1.130 Average

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:SCERror:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:SCERror:AVERage
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.scError.
↪average.fetch()
```

Return the symbol clock error results in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas. Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
clock_err: No help available

6.4.1.1.1.131 Current

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:SCERror:CURRent
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:SCERror:CURRent
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.scError.
↪current.fetch()
```

Return the symbol clock error results in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas. Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
clock_err: No help available

6.4.1.1.1.132 Maximum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:SCERror:MAXimum
```

class MaximumCls

Maximum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:SCERror:MAXimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.scError.
↪maximum.fetch()
```

Return the symbol clock error results in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas. Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
clock_err: No help available

6.4.1.1.1.133 Minimum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:SCERror:MINimum
```

class MinimumCls

Minimum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:SCERror:MINimum
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.scError.
↪minimum.fetch()
```

Return the symbol clock error results in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas. Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
clock_err: No help available

6.4.1.1.1.134 StandardDev

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:SCERror:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:SCERror:SDEViation
value: List[float] = driver.wlanMeas.multiEval.listPy.modulation.scError.
↪standardDev.fetch()
```

Return the symbol clock error results in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas. Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
clock_err: No help available

6.4.1.1.1.135 Scount

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:SCOunt
```

class ScountCls

Scount commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[int]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:SCOunt
value: List[int] = driver.wlanMeas.multiEval.listPy.modulation.scount.fetch()
```

Returns the expired statistic counts for modulation results over all segments in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
exp_stat_counts_mod: No help available

6.4.1.1.2 Segment<SegmentB>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.wlanMeas.multiEval.listPy.segment.repcap_segmentB_get()
driver.wlanMeas.multiEval.listPy.segment.repcap_segmentB_set(repcap.SegmentB.Nr1)
```

class SegmentCls

Segment commands group definition. 18 total commands, 2 Subgroups, 0 group commands Repeated Capability: SegmentB, default value after init: SegmentB.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.listPy.segment.clone()
```

Subgroups

6.4.1.1.2.1 Modulation

class ModulationCls

Modulation commands group definition. 10 total commands, 6 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.listPy.segment.modulation.clone()
```

Subgroups

6.4.1.1.2.2 Average

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>:MODulation:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Mcs_Index: int: Modulation and coding scheme index
- Mod_Type: enums.ModulationTypeD: Modulation scheme and coding rate UNSPecified: modulation unknown BPSK: BPSK, coding rate unknown BPSK12, BPSK34 (BPSKab) : BPSK, coding rate a/b BPSK14: BPSK, coding rate 1/2 DCM QPSK: QPSK, coding rate unknown QPSK12, QPSK34 (QPSKab) : QPSK, coding rate a/b QPSK14: QPSK, coding rate 1/2 DCM 16Q: 16QAM, coding rate unknown 16Q12, 16Q34 (16Qab) : 16QAM, coding rate a/b 16Q14: 16QAM, coding rate 1/2 DCM 16Q38: 16QAM, coding rate 3/4 DCM 64Q: 64QAM, coding rate unknown 64Q12, 64Q23, 64Q34, 64Q56 (64Qab) : 64QAM, coding rate a/b 256Q: 256QAM, coding rate unknown 256Q34, 256Q56 (256Qab) : 256QAM, coding rate a/b 1KQ: 1024QAM, coding rate unknown 1KQ34, 1KQ56 (1KQab) : 1024QAM, coding rate a/b BMCS14: BPSK DCM DUP BMCS15: BPSK DCM 4KQ: 4096QAM, coding rate unknown 4KQ34, 4KQ56 (4KQab) : 4096QAM, coding rate a/b
- Payload_Sym: int: Number of OFDM symbols in the payload of the measured burst
- Measured_Sym: int: The number of OFDM payload symbols to be measured.
- Payload_Bytes: int: Number of bytes in the payload of the measured burst.
- Guard_Interval: enums.GuardInterval: SHORt, LONG: short or long guard interval (up to 802.11ac) GI08, GI16, GI32: 0.8 s, 1.6 s, and 3.2 s guard interval durations (for 802.11ax)
- Nof_Ss: int: Number of spatial streams
- No_Of_Sts: int: Number of space-time streams
- Burst_Rate: float: The rate of bursts of the selected modulation format 5_ModType in the bursts received.
- Power_Backoff: float: Minimum distance of signal power to reference level since the start of the measurement.

- Burst_Power: float: RMS power of the measured burst
- Peak_Power: float: Peak power of the measured burst
- Crest_Factor: float: No parameter help available
- Evm_All_Carr: float: EVM for all, data, and pilot carriers
- Evm_Data_Carr: float: EVM for all, data, and pilot carriers
- Evm_Pilot_Carr: float: EVM for all, data, and pilot carriers
- Freq_Error: float: Center frequency error
- Clock_Error: float: Symbol clock error
- Iq_Offset: float: No parameter help available
- Dc_Power: float: No parameter help available
- Gain_Imbalance: float: No parameter help available
- Quad_Error: float: Quadrature error
- Ltf_Power: float: Power of long training fields (LTF) portion
- Data_Power: float: Power of data portion

fetch(*segmentB=SegmentB.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:MODulation:AVERage
value: FetchStruct = driver.wlanMeas.multiEval.listPy.segment.modulation.
↪average.fetch(segmentB = repcap.SegmentB.Default)
```

Return OFDM/OFDMA modulation single value results for segment <no> in list mode.

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.1.2.3 Current

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>:MODulation:CURRent
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.

- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Mcs_Index: int: Modulation and coding scheme index
- Mod_Type: enums.ModulationTypeD: Modulation scheme and coding rate UNSPecified: modulation unknown BPSK: BPSK, coding rate unknown BPSK12, BPSK34 (BPSKab) : BPSK, coding rate a/b BPSK14: BPSK, coding rate 1/2 DCM QPSK: QPSK, coding rate unknown QPSK12, QPSK34 (QPSKab) : QPSK, coding rate a/b QPSK14: QPSK, coding rate 1/2 DCM 16Q: 16QAM, coding rate unknown 16Q12, 16Q34 (16Qab) : 16QAM, coding rate a/b 16Q14: 16QAM, coding rate 1/2 DCM 16Q38: 16QAM, coding rate 3/4 DCM 64Q: 64QAM, coding rate unknown 64Q12, 64Q23, 64Q34, 64Q56 (64Qab) : 64QAM, coding rate a/b 256Q: 256QAM, coding rate unknown 256Q34, 256Q56 (256Qab) : 256QAM, coding rate a/b 1KQ: 1024QAM, coding rate unknown 1KQ34, 1KQ56 (1KQab) : 1024QAM, coding rate a/b BMCS14: BPSK DCM DUP BMCS15: BPSK DCM 4KQ: 4096QAM, coding rate unknown 4KQ34, 4KQ56 (4KQab) : 4096QAM, coding rate a/b
- Payload_Sym: int: Number of OFDM symbols in the payload of the measured burst
- Measured_Sym: int: The number of OFDM payload symbols to be measured.
- Payload_Bytes: int: Number of bytes in the payload of the measured burst.
- Guard_Interval: enums.GuardInterval: SHORt, LONG: short or long guard interval (up to 802.11ac) GI08, GI16, GI32: 0.8 s, 1.6 s, and 3.2 s guard interval durations (for 802.11ax)
- Nof_Ss: int: Number of spatial streams
- No_Of_Sts: int: Number of space-time streams
- Burst_Rate: float: The rate of bursts of the selected modulation format 5_ModType in the bursts received.
- Power_Backoff: float: Minimum distance of signal power to reference level since the start of the measurement.
- Burst_Power: float: RMS power of the measured burst
- Peak_Power: float: Peak power of the measured burst
- Crest_Factor: float: No parameter help available
- Evm_All_Carr: float: EVM for all, data, and pilot carriers
- Evm_Data_Carr: float: EVM for all, data, and pilot carriers
- Evm_Pilot_Carr: float: EVM for all, data, and pilot carriers
- Freq_Error: float: Center frequency error
- Clock_Error: float: Symbol clock error
- Iq_Offset: float: No parameter help available
- Dc_Power: float: No parameter help available
- Gain_Imbalance: float: No parameter help available
- Quad_Error: float: Quadrature error
- Ltf_Power: float: Power of long training fields (LTF) portion
- Data_Power: float: Power of data portion

fetch(*segmentB=SegmentB.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:MODulation:CURRent
value: FetchStruct = driver.wlanMeas.multiEval.listPy.segment.modulation.
↪current.fetch(segmentB = repcap.SegmentB.Default)
```

Return OFDM/OFDMA modulation single value results for segment <no> in list mode.

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.1.2.4 Dsss

class DsssCls

Dsss commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.listPy.segment.modulation.dsss.clone()
```

Subgroups

6.4.1.1.2.5 Average

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:MODulation:DSSS:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Seg_Reliability: int: Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Mod_Type: enums.ModulationTypeC: Modulation scheme and coding rate DBPSk1: 1 Mbit/s DBPSK DQPSk2: 2 Mbit/s DQPSK CCK5: 5.5 Mbit/s CCK CCK11: 11 Mbit/s CCK

- Plcp_Type: enums.PlcpType: Short or long PLCP
- Payload_Length: int: Number of bytes in the payload of the measured burst
- Burst_Power: float: RMS power of the measured burst
- Evm_Peak: float: Error vector magnitude peak value
- Evm_Rms: float: Error vector magnitude RMS value
- Freq_Error: float: Center frequency error
- Clock_Error: float: Chip clock error
- Iq_Offset: float: No parameter help available
- Gain_Imbalance: float: Gain imbalance
- Quad_Error: float: Quadrature error

fetch(*segmentB=SegmentB.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:MODulation:DSSS:AVERage
value: FetchStruct = driver.wlanMeas.multiEval.listPy.segment.modulation.dsss.
↪average.fetch(segmentB = repcap.SegmentB.Default)
```

Return the current, average, minimum, maximum and standard deviation single value results for DSSS signals for segment <no> in list mode.

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.1.2.6 Current

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:MODulation:DSSS:CURRent
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Seg_Reliability: int: Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.

- Mod_Type: enums.ModulationTypeC: Modulation scheme and coding rate DBPSk1: 1 Mbit/s DBPSK DQPSk2: 2 Mbit/s DQPSK CCK5: 5.5 Mbit/s CCK CCK11: 11 Mbit/s CCK
- Plcp_Type: enums.PlcpType: Short or long PLCP
- Payload_Length: int: Number of bytes in the payload of the measured burst
- Burst_Power: float: RMS power of the measured burst
- Evm_Peak: float: Error vector magnitude peak value
- Evm_Rms: float: Error vector magnitude RMS value
- Freq_Error: float: Center frequency error
- Clock_Error: float: Chip clock error
- Iq_Offset: float: No parameter help available
- Gain_Imbalance: float: Gain imbalance
- Quad_Error: float: Quadrature error

fetch(*segmentB=SegmentB.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:MODulation:DSSS:CURRent
value: FetchStruct = driver.wlanMeas.multiEval.listPy.segment.modulation.dsss.
↪current.fetch(segmentB = repcap.SegmentB.Default)
```

Return the current, average, minimum, maximum and standard deviation single value results for DSSS signals for segment <no> in list mode.

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.1.2.7 Maximum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:MODulation:DSSS:MAXimum
```

class MaximumCls

Maximum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Seg_Reliability: int: Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.

- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Mod_Type: enums.ModulationTypeC: Modulation scheme and coding rate DBPSk1: 1 Mbit/s DBPSK DQPSk2: 2 Mbit/s DQPSK CCK5: 5.5 Mbit/s CCK CCK11: 11 Mbit/s CCK
- Plcp_Type: enums.PlcpType: Short or long PLCP
- Payload_Length: int: Number of bytes in the payload of the measured burst
- Burst_Power: float: RMS power of the measured burst
- Evm_Peak: float: Error vector magnitude peak value
- Evm_Rms: float: Error vector magnitude RMS value
- Freq_Error: float: Center frequency error
- Clock_Error: float: Chip clock error
- Iq_Offset: float: No parameter help available
- Gain_Imbalance: float: Gain imbalance
- Quad_Error: float: Quadrature error

fetch(*segmentB=SegmentB.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:MODulation:DSSS:MAXimum
value: FetchStruct = driver.wlanMeas.multiEval.listPy.segment.modulation.dsss.
↪maximum.fetch(segmentB = repcap.SegmentB.Default)
```

Return the current, average, minimum, maximum and standard deviation single value results for DSSS signals for segment <no> in list mode.

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.1.2.8 Minimum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:MODulation:DSSS:MINimum
```

class MinimumCls

Minimum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.

- Seg_Reliability: int: Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Mod_Type: enums.ModulationTypeC: Modulation scheme and coding rate DBPSk1: 1 Mbit/s DBPSK DQPSk2: 2 Mbit/s DQPSK CCK5: 5.5 Mbit/s CCK CCK11: 11 Mbit/s CCK
- Plcp_Type: enums.PlcpType: Short or long PLCP
- Payload_Length: int: Number of bytes in the payload of the measured burst
- Burst_Power: float: RMS power of the measured burst
- Evm_Peak: float: Error vector magnitude peak value
- Evm_Rms: float: Error vector magnitude RMS value
- Freq_Error: float: Center frequency error
- Clock_Error: float: Chip clock error
- Iq_Offset: float: No parameter help available
- Gain_Imbalance: float: Gain imbalance
- Quad_Error: float: Quadrature error

fetch(*segmentB=SegmentB.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:MODulation:DSSS:MINimum
value: FetchStruct = driver.wlanMeas.multiEval.listPy.segment.modulation.dsss.
↪minimum.fetch(segmentB = repcap.SegmentB.Default)
```

Return the current, average, minimum, maximum and standard deviation single value results for DSSS signals for segment <no> in list mode.

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.1.2.9 StandardDev

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:MODulation:DSSS:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Seg_Reliability: int: Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Mod_Type: enums.ModulationTypeC: Modulation scheme and coding rate DBPSk1: 1 Mbit/s DBPSK DQPSk2: 2 Mbit/s DQPSK CCK5: 5.5 Mbit/s CCK CCK11: 11 Mbit/s CCK
- Plcp_Type: enums.PlcpType: Short or long PLCP
- Payload_Length: int: Number of bytes in the payload of the measured burst
- Burst_Power: float: RMS power of the measured burst
- Evm_Peak: float: Error vector magnitude peak value
- Evm_Rms: float: Error vector magnitude RMS value
- Freq_Error: float: Center frequency error
- Clock_Error: float: Chip clock error
- Iq_Offset: float: No parameter help available
- Gain_Imbalance: float: Gain imbalance
- Quad_Error: float: Quadrature error

fetch(*segmentB=SegmentB.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:MODulation:DSSS:SDEViation
value: FetchStruct = driver.wlanMeas.multiEval.listPy.segment.modulation.dsss.
↪standardDev.fetch(segmentB = repcap.SegmentB.Default)
```

Return the current, average, minimum, maximum and standard deviation single value results for DSSS signals for segment <no> in list mode.

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.1.2.10 Maximum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>:MODulation:MAXimum
```

class MaximumCls

Maximum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

`class FetchStruct`

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Mcs_Index: int: Modulation and coding scheme index
- Mod_Type: enums.ModulationTypeD: Modulation scheme and coding rate UNSPecified: modulation unknown BPSK: BPSK, coding rate unknown BPSK12, BPSK34 (BPSKab) : BPSK, coding rate a/b BPSK14: BPSK, coding rate 1/2 DCM QPSK: QPSK, coding rate unknown QPSK12, QPSK34 (QPSKab) : QPSK, coding rate a/b QPSK14: QPSK, coding rate 1/2 DCM 16Q: 16QAM, coding rate unknown 16Q12, 16Q34 (16Qab) : 16QAM, coding rate a/b 16Q14: 16QAM, coding rate 1/2 DCM 16Q38: 16QAM, coding rate 3/4 DCM 64Q: 64QAM, coding rate unknown 64Q12, 64Q23, 64Q34, 64Q56 (64Qab) : 64QAM, coding rate a/b 256Q: 256QAM, coding rate unknown 256Q34, 256Q56 (256Qab) : 256QAM, coding rate a/b 1KQ: 1024QAM, coding rate unknown 1KQ34, 1KQ56 (1KQab) : 1024QAM, coding rate a/b BMCS14: BPSK DCM DUP BMCS15: BPSK DCM 4KQ: 4096QAM, coding rate unknown 4KQ34, 4KQ56 (4KQab) : 4096QAM, coding rate a/b
- Payload_Sym: int: Number of OFDM symbols in the payload of the measured burst
- Measured_Sym: int: The number of OFDM payload symbols to be measured.
- Payload_Bytes: int: Number of bytes in the payload of the measured burst.
- Guard_Interval: enums.GuardInterval: SHORt, LONG: short or long guard interval (up to 802.11ac) GI08, GI16, GI32: 0.8 s, 1.6 s, and 3.2 s guard interval durations (for 802.11ax)
- Nof_Ss: int: Number of spatial streams
- No_Of_Sts: int: Number of space-time streams
- Burst_Rate: float: The rate of bursts of the selected modulation format 5_ModType in the bursts received.
- Power_Backoff: float: Minimum distance of signal power to reference level since the start of the measurement.
- Burst_Power: float: RMS power of the measured burst
- Peak_Power: float: Peak power of the measured burst
- Crest_Factor: float: No parameter help available
- Evm_All_Carr: float: EVM for all, data, and pilot carriers
- Evm_Data_Carr: float: EVM for all, data, and pilot carriers
- Evm_Pilot_Carr: float: EVM for all, data, and pilot carriers
- Freq_Error: float: Center frequency error
- Clock_Error: float: Symbol clock error
- Iq_Offset: float: No parameter help available
- Dc_Power: float: No parameter help available
- Gain_Imbalance: float: No parameter help available
- Quad_Error: float: Quadrature error

- Ltf_Power: float: Power of long training fields (LTF) portion
- Data_Power: float: Power of data portion

fetch(*segmentB=SegmentB.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:MODulation:MAXimum
value: FetchStruct = driver.wlanMeas.multiEval.listPy.segment.modulation.
↪maximum.fetch(segmentB = repcap.SegmentB.Default)
```

Return OFDM/OFDMA modulation single value results for segment <no> in list mode.

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.1.2.11 Minimum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>:MODulation:MINimum
```

class MinimumCls

Minimum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Mcs_Index: int: Modulation and coding scheme index
- Mod_Type: enums.ModulationTypeD: Modulation scheme and coding rate UNSPecified: modulation unknown BPSK: BPSK, coding rate unknown BPSK12, BPSK34 (BPSKab) : BPSK, coding rate a/b BPSK14: BPSK, coding rate 1/2 DCM QPSK: QPSK, coding rate unknown QPSK12, QPSK34 (QPSKab) : QPSK, coding rate a/b QPSK14: QPSK, coding rate 1/2 DCM 16Q: 16QAM, coding rate unknown 16Q12, 16Q34 (16Qab) : 16QAM, coding rate a/b 16Q14: 16QAM, coding rate 1/2 DCM 16Q38: 16QAM, coding rate 3/4 DCM 64Q: 64QAM, coding rate unknown 64Q12, 64Q23, 64Q34, 64Q56 (64Qab) : 64QAM, coding rate a/b 256Q: 256QAM, coding rate unknown 256Q34, 256Q56 (256Qab) : 256QAM, coding rate a/b 1KQ: 1024QAM, coding rate unknown 1KQ34, 1KQ56 (1KQab) : 1024QAM, coding rate a/b BMCS14: BPSK DCM DUP BMCS15: BPSK DCM 4KQ: 4096QAM, coding rate unknown 4KQ34, 4KQ56 (4KQab) : 4096QAM, coding rate a/b
- Payload_Sym: int: Number of OFDM symbols in the payload of the measured burst
- Measured_Sym: int: The number of OFDM payload symbols to be measured.
- Payload_Bytes: int: Number of bytes in the payload of the measured burst.

- Guard_Interval: enums.GuardInterval: SHORt, LONG: short or long guard interval (up to 802.11ac) GI08, GI16, GI32: 0.8 s, 1.6 s, and 3.2 s guard interval durations (for 802.11ax)
- Nof_Ss: int: Number of spatial streams
- No_Of_Sts: int: Number of space-time streams
- Burst_Rate: float: The rate of bursts of the selected modulation format 5_ModType in the bursts received.
- Power_Backoff: float: Minimum distance of signal power to reference level since the start of the measurement.
- Burst_Power: float: RMS power of the measured burst
- Peak_Power: float: Peak power of the measured burst
- Crest_Factor: float: No parameter help available
- Evm_All_Carr: float: EVM for all, data, and pilot carriers
- Evm_Data_Carr: float: EVM for all, data, and pilot carriers
- Evm_Pilot_Carr: float: EVM for all, data, and pilot carriers
- Freq_Error: float: Center frequency error
- Clock_Error: float: Symbol clock error
- Iq_Offset: float: No parameter help available
- Dc_Power: float: No parameter help available
- Gain_Imbalance: float: No parameter help available
- Quad_Error: float: Quadrature error
- Ltf_Power: float: Power of long training fields (LTF) portion
- Data_Power: float: Power of data portion

fetch(*segmentB=SegmentB.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:MODulation:MINimum
value: FetchStruct = driver.wlanMeas.multiEval.listPy.segment.modulation.
↪minimum.fetch(segmentB = repcap.SegmentB.Default)
```

Return OFDM/OFDMA modulation single value results for segment <no> in list mode.

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.1.2.12 StandardDev

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>:MODulation:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Mcs_Index: int: Modulation and coding scheme index
- Mod_Type: enums.ModulationTypeD: Modulation scheme and coding rate UNSPecified: modulation unknown BPSK: BPSK, coding rate unknown BPSK12, BPSK34 (BPSKab) : BPSK, coding rate a/b BPSK14: BPSK, coding rate 1/2 DCM QPSK: QPSK, coding rate unknown QPSK12, QPSK34 (QPSKab) : QPSK, coding rate a/b QPSK14: QPSK, coding rate 1/2 DCM 16Q: 16QAM, coding rate unknown 16Q12, 16Q34 (16Qab) : 16QAM, coding rate a/b 16Q14: 16QAM, coding rate 1/2 DCM 16Q38: 16QAM, coding rate 3/4 DCM 64Q: 64QAM, coding rate unknown 64Q12, 64Q23, 64Q34, 64Q56 (64Qab) : 64QAM, coding rate a/b 256Q: 256QAM, coding rate unknown 256Q34, 256Q56 (256Qab) : 256QAM, coding rate a/b 1KQ: 1024QAM, coding rate unknown 1KQ34, 1KQ56 (1KQab) : 1024QAM, coding rate a/b BMCS14: BPSK DCM DUP BMCS15: BPSK DCM 4KQ: 4096QAM, coding rate unknown 4KQ34, 4KQ56 (4KQab) : 4096QAM, coding rate a/b
- Payload_Sym: int: Number of OFDM symbols in the payload of the measured burst
- Measured_Sym: int: The number of OFDM payload symbols to be measured.
- Payload_Bytes: int: Number of bytes in the payload of the measured burst.
- Guard_Interval: enums.GuardInterval: SHORt, LONG: short or long guard interval (up to 802.11ac) GI08, GI16, GI32: 0.8 s, 1.6 s, and 3.2 s guard interval durations (for 802.11ax)
- Nof_Ss: int: Number of spatial streams
- No_Of_Sts: int: Number of space-time streams
- Burst_Rate: float: The rate of bursts of the selected modulation format 5_ModType in the bursts received.
- Power_Backoff: float: Minimum distance of signal power to reference level since the start of the measurement.
- Burst_Power: float: RMS power of the measured burst
- Peak_Power: float: Peak power of the measured burst
- Crest_Factor: float: No parameter help available
- Evm_All_Carr: float: EVM for all, data, and pilot carriers
- Evm_Data_Carr: float: EVM for all, data, and pilot carriers
- Evm_Pilot_Carr: float: EVM for all, data, and pilot carriers

- Freq_Error: float: Center frequency error
- Clock_Error: float: Symbol clock error
- Iq_Offset: float: No parameter help available
- Dc_Power: float: No parameter help available
- Gain_Imbalance: float: No parameter help available
- Quad_Error: float: Quadrature error
- Ltf_Power: float: Power of long training fields (LTF) portion
- Data_Power: float: Power of data portion

fetch(*segmentB=SegmentB.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:MODulation:SDEViation
value: FetchStruct = driver.wlanMeas.multiEval.listPy.segment.modulation.
↪standardDev.fetch(segmentB = repcap.SegmentB.Default)
```

Return OFDM/OFDMA modulation single value results for segment <no> in list mode.

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.1.2.13 TsMask

class TsMaskCls

TsMask commands group definition. 8 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.listPy.segment.tsMask.clone()
```

Subgroups

6.4.1.1.2.14 Average

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>:TSMask:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Seg_Reliability: int: Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Out_Of_Tol: float: Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified transmit spectrum mask limits.
- Margins: List[float]: Comma-separated list of margin values, one value per spectrum mask area The number of margin values depends on the selected standard, see Table 'Spectrum mask areas'.

fetch(*segmentB=SegmentB.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:TSMask:AVERage
value: FetchStruct = driver.wlanMeas.multiEval.listPy.segment.tsMask.average.
↪fetch(segmentB = repcap.SegmentB.Default)
```

Return limit line margin results for segment <no> in list mode. A positive result indicates that the trace is located above the limit line, i.e. the limit is exceeded. Margins for the current, average, minimum, maximum and standard deviation results are returned.

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.1.2.15 Current

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>:TSMask:CURRent
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Seg_Reliability: int: Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Out_Of_Tol: float: Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified transmit spectrum mask limits.
- Margins: List[float]: Comma-separated list of margin values, one value per spectrum mask area The number of margin values depends on the selected standard, see Table 'Spectrum mask areas'.

fetch(*segmentB=SegmentB.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:TSMask:CURRent
value: FetchStruct = driver.wlanMeas.multiEval.listPy.segment.tsMask.current.
↪fetch(segmentB = repcap.SegmentB.Default)
```

Return limit line margin results for segment <no> in list mode. A positive result indicates that the trace is located above the limit line, i.e. the limit is exceeded. Margins for the current, average, minimum, maximum and standard deviation results are returned.

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.1.2.16 Frequency

class FrequencyCls

Frequency commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.listPy.segment.tsMask.frequency.clone()
```

Subgroups

6.4.1.1.2.17 Average

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:TSMask:FREQuency:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Seg_Reliability: int: Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Out_Of_Tol: float: Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified transmit spectrum mask limits.

- Frequencies: List[float]: Comma-separated list of frequencies, one value per margin The number of margins equals the number of spectrum mask areas and depends on the selected standard, see Table 'Spectrum mask areas'.

fetch(*segmentB=SegmentB.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:TSMask:FREQuency:AVERage
value: FetchStruct = driver.wlanMeas.multiEval.listPy.segment.tsMask.frequency.
↪average.fetch(segmentB = repcap.SegmentB.Default)
```

Return the X-positions of the limit line margins of the transmit spectrum mask for list mode, segment <no>. Positions for the current, average, minimum, maximum and standard deviation results are returned.

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.1.2.18 Current

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:TSMask:FREQuency:CURRent
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Seg_Reliability: int: Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Out_Of_Tol: float: Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified transmit spectrum mask limits.
- Frequencies: List[float]: Comma-separated list of frequencies, one value per margin The number of margins equals the number of spectrum mask areas and depends on the selected standard, see Table 'Spectrum mask areas'.

fetch(*segmentB=SegmentB.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:TSMask:FREQuency:CURRent
value: FetchStruct = driver.wlanMeas.multiEval.listPy.segment.tsMask.frequency.
↪current.fetch(segmentB = repcap.SegmentB.Default)
```

Return the X-positions of the limit line margins of the transmit spectrum mask for list mode, segment <no>. Positions for the current, average, minimum, maximum and standard deviation results are returned.

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.1.2.19 Maximum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:TSMask:FREQuency:MAXimum
```

class MaximumCls

Maximum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Seg_Reliability: int: Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Out_Of_Tol: float: Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified transmit spectrum mask limits.
- Frequencies: List[float]: Comma-separated list of frequencies, one value per margin The number of margins equals the number of spectrum mask areas and depends on the selected standard, see Table 'Spectrum mask areas'.

fetch(*segmentB=SegmentB.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:TSMask:FREQuency:MAXimum
value: FetchStruct = driver.wlanMeas.multiEval.listPy.segment.tsMask.frequency.
↪maximum.fetch(segmentB = repcap.SegmentB.Default)
```

Return the X-positions of the limit line margins of the transmit spectrum mask for list mode, segment <no>. Positions for the current, average, minimum, maximum and standard deviation results are returned.

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.1.2.20 Minimum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:TSMask:FREQuency:MINimum
```

class MinimumCls

Minimum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Seg_Reliability: int: Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Out_Of_Tol: float: Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified transmit spectrum mask limits.
- Frequencies: List[float]: Comma-separated list of frequencies, one value per margin The number of margins equals the number of spectrum mask areas and depends on the selected standard, see Table 'Spectrum mask areas'.

fetch(*segmentB=SegmentB.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:TSMask:FREQuency:MINimum
value: FetchStruct = driver.wlanMeas.multiEval.listPy.segment.tsMask.frequency.
↪minimum.fetch(segmentB = repcap.SegmentB.Default)
```

Return the X-positions of the limit line margins of the transmit spectrum mask for list mode, segment <no>. Positions for the current, average, minimum, maximum and standard deviation results are returned.

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.1.2.21 Maximum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>:TSMask:MAXimum
```

class MaximumCls

Maximum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Seg_Reliability: int: Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Out_Of_Tol: float: Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified transmit spectrum mask limits.
- Margins: List[float]: Comma-separated list of margin values, one value per spectrum mask area The number of margin values depends on the selected standard, see Table 'Spectrum mask areas'.

fetch(*segmentB=SegmentB.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:TSMask:MAXimum
value: FetchStruct = driver.wlanMeas.multiEval.listPy.segment.tsMask.maximum.
↪fetch(segmentB = repcap.SegmentB.Default)
```

Return limit line margin results for segment <no> in list mode. A positive result indicates that the trace is located above the limit line, i.e. the limit is exceeded. Margins for the current, average, minimum, maximum and standard deviation results are returned.

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.1.2.22 Minimum

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>:TSMask:MINimum
```

class MinimumCls

Minimum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Seg_Reliability: int: Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Out_Of_Tol: float: Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified transmit spectrum mask limits.
- Margins: List[float]: Comma-separated list of margin values, one value per spectrum mask area The number of margin values depends on the selected standard, see Table 'Spectrum mask areas'.

fetch(*segmentB=SegmentB.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<segment>
↪:TSMask:MINimum
value: FetchStruct = driver.wlanMeas.multiEval.listPy.segment.tsMask.minimum.
↪fetch(segmentB = repcap.SegmentB.Default)
```

Return limit line margin results for segment <no> in list mode. A positive result indicates that the trace is located above the limit line, i.e. the limit is exceeded. Margins for the current, average, minimum, maximum and standard deviation results are returned.

param segmentB
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.1.3 Sreliability

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SRELiability
```

class SreliabilityCls

Sreliability commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[int]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SRELiability
value: List[int] = driver.wlanMeas.multiEval.listPy.sreliability.fetch()
```

Returns the segment reliability for all measured list mode segments. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
seg_reliabilities: Comma-separated list of n values, one per measured segment The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.

6.4.1.1.4 TsMask

class TsMaskCls

TsMask commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.listPy.tsMask.clone()
```

Subgroups

6.4.1.1.4.1 Scount

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:TSMask:SCOunt
```

class ScountCls

Scount commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[int]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:TSMask:SCOunt
value: List[int] = driver.wlanMeas.multiEval.listPy.tsMask.scount.fetch()
```

Returns the expired statistic counts for transmit spectrum mask results over all segments in list mode. The values in curly brackets {} are specified for each active segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The number of active segments n is determined by method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.ListPy.count.

Suppressed linked return values: reliability

return
exp_stat_counts_tsm: No help available

6.4.1.2 Modulation

class ModulationCls

Modulation commands group definition. 144 total commands, 14 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.modulation.clone()
```

Subgroups

6.4.1.2.1 Acsiso

class AcsisoCls

Acsiso commands group definition. 12 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.modulation.acsiso.clone()
```

Subgroups

6.4.1.2.1.1 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:ACSiso:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:ACSiso:AVERage
CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:ACSiso:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Mcs_Index: enums.ResultStatus2: No parameter help available
- Payload_Length: enums.ResultStatus2: No parameter help available
- Guard_Interval: enums.ResultStatus2: No parameter help available
- Burst_Power: enums.ResultStatus2: No parameter help available
- Evm_All_Carr: enums.ResultStatus2: No parameter help available
- Evm_Data_Carr: enums.ResultStatus2: No parameter help available
- Evm_Pilot_Carr: enums.ResultStatus2: No parameter help available
- Freq_Error: enums.ResultStatus2: No parameter help available
- Clock_Error: enums.ResultStatus2: No parameter help available
- Iq_Offset: enums.ResultStatus2: No parameter help available
- Out_Of_Tol: enums.ResultStatus2: No parameter help available
- Iq_Offset_8080: enums.ResultStatus2: No parameter help available
- Gain_Imbal: enums.ResultStatus2: No parameter help available
- Quad_Error: enums.ResultStatus2: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Mcs_Index: int: No parameter help available
- Payload_Length: int: No parameter help available
- Guard_Interval: enums.GuardInterval: No parameter help available
- Burst_Power: float: No parameter help available
- Evm_All_Carr: float: No parameter help available
- Evm_Data_Carr: float: No parameter help available
- Evm_Pilot_Carr: float: No parameter help available
- Freq_Error: float: No parameter help available
- Clock_Error: float: No parameter help available
- Iq_Offset: float: No parameter help available
- Out_Of_Tol: float: No parameter help available
- Iq_Offset_8080: float: No parameter help available
- Gain_Imbal: float: No parameter help available
- Quad_Error: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>
↪:MEValuation:MODulation:ACSiso:AVERage
value: CalculateStruct = driver.wlanMeas.multiEval.modulation.acsiso.average.
↪calculate()
```

No command help available

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:ACSiso:AVERage
value: ResultData = driver.wlanMeas.multiEval.modulation.acsiso.average.fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:ACSiso:AVERage
value: ResultData = driver.wlanMeas.multiEval.modulation.acsiso.average.read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.2.1.2 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:ACSiso:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:ACSiso:CURRent
CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:ACSiso:CURRent
```

class CurrentCls

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Mcs_Index: enums.ResultStatus2: No parameter help available
- Payload_Length: enums.ResultStatus2: No parameter help available
- Guard_Interval: enums.ResultStatus2: No parameter help available
- Burst_Power: enums.ResultStatus2: No parameter help available
- Evm_All_Carr: enums.ResultStatus2: No parameter help available
- Evm_Data_Carr: enums.ResultStatus2: No parameter help available
- Evm_Pilot_Carr: enums.ResultStatus2: No parameter help available
- Freq_Error: enums.ResultStatus2: No parameter help available
- Clock_Error: enums.ResultStatus2: No parameter help available
- Iq_Offset: enums.ResultStatus2: No parameter help available
- Out_Of_Tol: enums.ResultStatus2: No parameter help available
- Iq_Offset_8080: enums.ResultStatus2: No parameter help available
- Gain_Imbal: enums.ResultStatus2: No parameter help available
- Quad_Error: enums.ResultStatus2: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Mcs_Index: int: No parameter help available
- Payload_Length: int: No parameter help available
- Guard_Interval: enums.GuardInterval: No parameter help available
- Burst_Power: float: No parameter help available
- Evm_All_Carr: float: No parameter help available
- Evm_Data_Carr: float: No parameter help available
- Evm_Pilot_Carr: float: No parameter help available
- Freq_Error: float: No parameter help available
- Clock_Error: float: No parameter help available

- Iq_Offset: float: No parameter help available
- Out_Of_Tol: float: No parameter help available
- Iq_Offset_8080: float: No parameter help available
- Gain_Imbal: float: No parameter help available
- Quad_Error: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>
↪:MEValuation:MODulation:ACSiso:CURRent
value: CalculateStruct = driver.wlanMeas.multiEval.modulation.acsiso.current.
↪calculate()
```

No command help available

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:ACSiso:CURRent
value: ResultData = driver.wlanMeas.multiEval.modulation.acsiso.current.fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:ACSiso:CURRent
value: ResultData = driver.wlanMeas.multiEval.modulation.acsiso.current.read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.2.1.3 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:ACSiso:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:ACSiso:MAXimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:ACSiso:MAXimum
```

class MaximumCls

Maximum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Mcs_Index: enums.ResultStatus2: No parameter help available

- Payload_Length: enums.ResultStatus2: No parameter help available
- Guard_Interval: enums.ResultStatus2: No parameter help available
- Burst_Power: enums.ResultStatus2: No parameter help available
- Evm_All_Carr: enums.ResultStatus2: No parameter help available
- Evm_Data_Carr: enums.ResultStatus2: No parameter help available
- Evm_Pilot_Carr: enums.ResultStatus2: No parameter help available
- Freq_Error: enums.ResultStatus2: No parameter help available
- Clock_Error: enums.ResultStatus2: No parameter help available
- Iq_Offset: enums.ResultStatus2: No parameter help available
- Out_Of_Tol: enums.ResultStatus2: No parameter help available
- Iq_Offset_8080: enums.ResultStatus2: No parameter help available
- Gain_Imbal: enums.ResultStatus2: No parameter help available
- Quad_Error: enums.ResultStatus2: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Mcs_Index: int: No parameter help available
- Payload_Length: int: No parameter help available
- Guard_Interval: enums.GuardInterval: No parameter help available
- Burst_Power: float: No parameter help available
- Evm_All_Carr: float: No parameter help available
- Evm_Data_Carr: float: No parameter help available
- Evm_Pilot_Carr: float: No parameter help available
- Freq_Error: float: No parameter help available
- Clock_Error: float: No parameter help available
- Iq_Offset: float: No parameter help available
- Out_Of_Tol: float: No parameter help available
- Iq_Offset_8080: float: No parameter help available
- Gain_Imbal: float: No parameter help available
- Quad_Error: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>
↪:MEValuation:MODulation:ACSiso:MAXimum
value: CalculateStruct = driver.wlanMeas.multiEval.modulation.acsiso.maximum.
↪calculate()
```

No command help available

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:ACSiso:MAXimum
value: ResultData = driver.wlanMeas.multiEval.modulation.acsiso.maximum.fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:ACSiso:MAXimum
value: ResultData = driver.wlanMeas.multiEval.modulation.acsiso.maximum.read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.2.1.4 StandardDev

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:ACSiso:SDEViation
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:ACSiso:SDEViation
CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:ACSiso:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Mcs_Index: enums.ResultStatus2: No parameter help available
- Payload_Length: enums.ResultStatus2: No parameter help available
- Guard_Interval: enums.ResultStatus2: No parameter help available
- Burst_Power: enums.ResultStatus2: No parameter help available
- Evm_All_Carr: enums.ResultStatus2: No parameter help available
- Evm_Data_Carr: enums.ResultStatus2: No parameter help available
- Evm_Pilot_Carr: enums.ResultStatus2: No parameter help available
- Freq_Error: enums.ResultStatus2: No parameter help available
- Clock_Error: enums.ResultStatus2: No parameter help available
- Iq_Offset: enums.ResultStatus2: No parameter help available
- Out_Of_Tol: enums.ResultStatus2: No parameter help available

- Iq_Offset_8080: enums.ResultStatus2: No parameter help available
- Gain_Imbal: enums.ResultStatus2: No parameter help available
- Quad_Error: enums.ResultStatus2: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Mcs_Index: int: No parameter help available
- Payload_Length: int: No parameter help available
- Guard_Interval: enums.GuardInterval: No parameter help available
- Burst_Power: float: No parameter help available
- Evm_All_Carr: float: No parameter help available
- Evm_Data_Carr: float: No parameter help available
- Evm_Pilot_Carr: float: No parameter help available
- Freq_Error: float: No parameter help available
- Clock_Error: float: No parameter help available
- Iq_Offset: float: No parameter help available
- Out_Of_Tol: float: No parameter help available
- Iq_Offset_8080: float: No parameter help available
- Gain_Imbal: float: No parameter help available
- Quad_Error: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>
↪:MEValuation:MODulation:ACSiso:SDEViation
value: CalculateStruct = driver.wlanMeas.multiEval.modulation.acsiso.
↪standardDev.calculate()
```

No command help available

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:MODulation:ACSiso:SDEViation
value: ResultData = driver.wlanMeas.multiEval.modulation.acsiso.standardDev.
↪fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:ACSiso:SDEViation
value: ResultData = driver.wlanMeas.multiEval.modulation.acsiso.standardDev.
↪read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.2.2 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:AVERage
CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tol: enums.ResultStatus2: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Mcs_Index: enums.ResultStatus2: Modulation and coding scheme index
- Mod_Type: enums.ResultStatus2: No parameter help available
- Payload_Sym: enums.ResultStatus2: Number of OFDM symbols in the payload of the measured burst
- Measured_Sym: enums.ResultStatus2: Number of measured payload OFDM symbols
- Payload_Bytes: enums.ResultStatus2: Number of bytes in the payload of the measured burst.
- Guard_Interval: enums.ResultStatus2: SHORt, LONG: short or long guard interval (up to 802.11ac) GI08, GI16, GI32: 0.8 s, 1.6 s, and 3.2 s guard interval durations (for 802.11ax)
- Nof_Ss: enums.ResultStatus2: Number of spatial streams
- No_Of_Sts: enums.ResultStatus2: Number of space-time streams
- Burst_Rate: enums.ResultStatus2: If a modulation filter is used, the burst rate indicates the share of bursts of the selected modulation format in the bursts received. Otherwise, it returns 1.
- Power_Backoff: enums.ResultStatus2: Minimum distance of signal power to reference level since the start of the measurement.
- Burst_Power: enums.ResultStatus2: RMS power of the measured burst
- Peak_Power: enums.ResultStatus2: Peak power of the measured burst
- Crest_Factor: enums.ResultStatus2: No parameter help available
- Evm_All_Carr: enums.ResultStatus2: EVM for all carriers
- Evm_Data_Carr: enums.ResultStatus2: EVM for data carriers

- Evm_Pilot_Carr: enums.ResultStatus2: EVM for pilot carriers
- Freq_Error: enums.ResultStatus2: Center frequency error
- Clock_Error: enums.ResultStatus2: Symbol clock error
- Iq_Offset: enums.ResultStatus2: No parameter help available
- Dc_Power: enums.ResultStatus2: Power of the DC subcarriers
- Gain_Imbalance: enums.ResultStatus2: Gain imbalance cannot be calculated if the spectrum is not symmetrical, e.g. for HE_TB and HE_MU.
- Quad_Error: enums.ResultStatus2: Quadrature error cannot be calculated if the spectrum is not symmetrical, e.g. for HE_TB and HE_MU.
- Ltf_Power: enums.ResultStatus2: Power of long training fields (LTF) portion
- Data_Power: enums.ResultStatus2: Power of data portion
- Preamble_Power: enums.ResultStatus2: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Mcs_Index: int: Modulation and coding scheme index
- Mod_Type: enums.ModulationTypeD: No parameter help available
- Payload_Sym: int: Number of OFDM symbols in the payload of the measured burst
- Measured_Sym: int: Number of measured payload OFDM symbols
- Payload_Bytes: int: Number of bytes in the payload of the measured burst.
- Guard_Interval: enums.GuardInterval: SHORt, LONG: short or long guard interval (up to 802.11ac) GI08, GI16, GI32: 0.8 s, 1.6 s, and 3.2 s guard interval durations (for 802.11ax)
- Nof_Ss: int: Number of spatial streams
- No_Of_Sts: int: Number of space-time streams
- Burst_Rate: float: If a modulation filter is used, the burst rate indicates the share of bursts of the selected modulation format in the bursts received. Otherwise, it returns 1.
- Power_Backoff: float: Minimum distance of signal power to reference level since the start of the measurement.
- Burst_Power: float: RMS power of the measured burst
- Peak_Power: float: Peak power of the measured burst
- Crest_Factor: float: No parameter help available
- Evm_All_Carr: float: EVM for all carriers
- Evm_Data_Carr: float: EVM for data carriers
- Evm_Pilot_Carr: float: EVM for pilot carriers
- Freq_Error: float: Center frequency error
- Clock_Error: float: Symbol clock error

- Iq_Offset: float: No parameter help available
- Dc_Power: float: Power of the DC subcarriers
- Gain_Imbalance: float: Gain imbalance cannot be calculated if the spectrum is not symmetrical, e.g. for HE_TB and HE_MU.
- Quad_Error: float: Quadrature error cannot be calculated if the spectrum is not symmetrical, e.g. for HE_TB and HE_MU.
- Ltf_Power: float: Power of long training fields (LTF) portion
- Data_Power: float: Power of data portion
- Preamble_Power: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:AVERage
value: CalculateStruct = driver.wlanMeas.multiEval.modulation.average.
↪calculate()
```

Return the single value results for OFDM SISO measurements. For MIMO measurements, the stream/antenna-independent values are returned. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:AVERage
value: ResultData = driver.wlanMeas.multiEval.modulation.average.fetch()
```

Return the single value results for OFDM SISO measurements. For MIMO measurements, the stream/antenna-independent values are returned. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:AVERage
value: ResultData = driver.wlanMeas.multiEval.modulation.average.read()
```

Return the single value results for OFDM SISO measurements. For MIMO measurements, the stream/antenna-independent values are returned. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.2.3 CfoDistribution

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:CFDistrib
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CFDistrib
CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:CFDistrib
```

class CfoDistributionCls

CfoDistribution commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Cfo_Percentage: float: Percentage of CFO errors
- Cfo_Outside: int: Number of detected CFO errors
- Cfo_Total: int: Number of measured CFOs

calculate() → ResultStatus2

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:CFDistrib
value: enums.ResultStatus2 = driver.wlanMeas.multiEval.modulation.
↪cfoDistribution.calculate()
```

Return the scalar results for carrier frequency offset (CFO) error distribution. The results are only supported for Wi-Fi 6 (802.11ax) and higher. Exceeding the limit has no impact on the stop On Limit Failure condition or out of tolerance counter.

Suppressed linked return values: reliability

return
cfo_percentage: Percentage of CFO errors

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CFDistrib
value: ResultData = driver.wlanMeas.multiEval.modulation.cfoDistribution.fetch()
```

Return the scalar results for carrier frequency offset (CFO) error distribution. The results are only supported for Wi-Fi 6 (802.11ax) and higher. Exceeding the limit has no impact on the stop On Limit Failure condition or out of tolerance counter.

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:CFDistrib
value: ResultData = driver.wlanMeas.multiEval.modulation.cfoDistribution.read()
```

Return the scalar results for carrier frequency offset (CFO) error distribution. The results are only supported for Wi-Fi 6 (802.11ax) and higher. Exceeding the limit has no impact on the stop On Limit Failure condition or out of tolerance counter.

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.2.4 Cmimo

class CmimoCls

Cmimo commands group definition. 18 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.modulation.cmimo.clone()
```

Subgroups

6.4.1.2.4.1 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:CMIMo:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CMIMo:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Mcs_Index: int: No parameter help available
- Nof_Ss: int: No parameter help available
- No_Of_Sts: int: No parameter help available
- Payload_Length: int: No parameter help available
- Evm_All_Carr: float: No parameter help available
- Evm_Data_Carr: float: No parameter help available
- Evm_Pilot_Carr: float: No parameter help available
- Power_Total: float: No parameter help available
- Power_Total_Peak: float: No parameter help available
- Power_Sts_1: float: No parameter help available
- Power_Sts_2: float: No parameter help available
- Power_Sts_3: float: No parameter help available
- Power_Sts_4: float: No parameter help available
- Freq_Error: float: No parameter help available

- Out_Of_Tol: float: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CMIMo:AVERage
value: ResultData = driver.wlanMeas.multiEval.modulation.cmimo.average.fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:CMIMo:AVERage
value: ResultData = driver.wlanMeas.multiEval.modulation.cmimo.average.read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.2.4.2 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:CMIMo:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CMIMo:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Mcs_Index: int: No parameter help available
- Nof_Ss: int: No parameter help available
- No_Of_Sts: int: No parameter help available
- Payload_Length: int: No parameter help available
- Evm_All_Carr: float: No parameter help available
- Evm_Data_Carr: float: No parameter help available
- Evm_Pilot_Carr: float: No parameter help available
- Power_Total: float: No parameter help available
- Power_Total_Peak: float: No parameter help available
- Power_Sts_1: float: No parameter help available
- Power_Sts_2: float: No parameter help available
- Power_Sts_3: float: No parameter help available
- Power_Sts_4: float: No parameter help available

- Freq_Error: float: No parameter help available
- Out_Of_Tol: float: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CMIMo:CURRent
value: ResultData = driver.wlanMeas.multiEval.modulation.cmimo.current.fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:CMIMo:CURRent
value: ResultData = driver.wlanMeas.multiEval.modulation.cmimo.current.read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.2.4.3 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:CMIMo:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CMIMo:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Mcs_Index: int: No parameter help available
- Nof_Ss: int: No parameter help available
- No_Of_Sts: int: No parameter help available
- Payload_Length: int: No parameter help available
- Evm_All_Carr: float: No parameter help available
- Evm_Data_Carr: float: No parameter help available
- Evm_Pilot_Carr: float: No parameter help available
- Power_Total: float: No parameter help available
- Power_Total_Peak: float: No parameter help available
- Power_Sts_1: float: No parameter help available
- Power_Sts_2: float: No parameter help available
- Power_Sts_3: float: No parameter help available

- Power_Sts_4: float: No parameter help available
- Freq_Error: float: No parameter help available
- Out_Of_Tol: float: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CMIMo:MAXimum
value: ResultData = driver.wlanMeas.multiEval.modulation.cmimo.maximum.fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:CMIMo:MAXimum
value: ResultData = driver.wlanMeas.multiEval.modulation.cmimo.maximum.read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.2.4.4 Psts

class PstsCls

Psts commands group definition. 10 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.modulation.cmimo.psts.clone()
```

Subgroups

6.4.1.2.4.5 Average

SCPI Commands :

```
READ:WLAN:MEASurement<instance>:MEValuation:MODulation:CMIMo:PSTS:AVERage
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:CMIMo:PSTS:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<instance>
↪:MEValuation:MODulation:CMIMo:PSTS:AVERage
value: List[float] = driver.wlanMeas.multiEval.modulation.cmimo.psts.average.
↪fetch()
```

Return the single value RMS power results for the individual space-time streams. The current, average, minimum, maximum, and standard deviation results can be retrieved. For a meaningful result, set the spatial mapping matrix in the DUT to direct mapping. It causes a one-to-one mapping of space time streams to TX antennas. Thus a broken TX chain (no power) is detected and a damaged chain is identified by its bad EVM.

Suppressed linked return values: reliability

return
power_sts_tx: Four values, one value per space-time stream

read() → List[float]

```
# SCPI: READ:WLAN:MEASurement<instance>
↪:MEValuation:MODulation:CMIMo:PSTS:AVERage
value: List[float] = driver.wlanMeas.multiEval.modulation.cmimo.psts.average.
↪read()
```

Return the single value RMS power results for the individual space-time streams. The current, average, minimum, maximum, and standard deviation results can be retrieved. For a meaningful result, set the spatial mapping matrix in the DUT to direct mapping. It causes a one-to-one mapping of space time streams to TX antennas. Thus a broken TX chain (no power) is detected and a damaged chain is identified by its bad EVM.

Suppressed linked return values: reliability

return
power_sts_tx: Four values, one value per space-time stream

6.4.1.2.4.6 Current

SCPI Commands :

```
READ:WLAN:MEASurement<instance>:MEValuation:MODulation:CMIMo:PSTS:CURRent
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:CMIMo:PSTS:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<instance>
↪:MEValuation:MODulation:CMIMo:PSTS:CURRent
value: List[float] = driver.wlanMeas.multiEval.modulation.cmimo.psts.current.
↪fetch()
```

Return the single value RMS power results for the individual space-time streams. The current, average, minimum, maximum, and standard deviation results can be retrieved. For a meaningful result, set the spatial mapping matrix in the DUT to direct mapping. It causes a one-to-one mapping of space time streams to

TX antennas. Thus a broken TX chain (no power) is detected and a damaged chain is identified by its bad EVM.

Suppressed linked return values: reliability

return
power_sts_tx: Four values, one value per space-time stream

read() → List[float]

```
# SCPI: READ:WLAN:MEASurement<instance>
↪:MEValuation:MODulation:CMIMo:PSTS:CURRent
value: List[float] = driver.wlanMeas.multiEval.modulation.cmimo.psts.current.
↪read()
```

Return the single value RMS power results for the individual space-time streams. The current, average, minimum, maximum, and standard deviation results can be retrieved. For a meaningful result, set the spatial mapping matrix in the DUT to direct mapping. It causes a one-to-one mapping of space time streams to TX antennas. Thus a broken TX chain (no power) is detected and a damaged chain is identified by its bad EVM.

Suppressed linked return values: reliability

return
power_sts_tx: Four values, one value per space-time stream

6.4.1.2.4.7 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<instance>:MEValuation:MODulation:CMIMo:PSTS:MAXimum
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:CMIMo:PSTS:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<instance>
↪:MEValuation:MODulation:CMIMo:PSTS:MAXimum
value: List[float] = driver.wlanMeas.multiEval.modulation.cmimo.psts.maximum.
↪fetch()
```

Return the single value RMS power results for the individual space-time streams. The current, average, minimum, maximum, and standard deviation results can be retrieved. For a meaningful result, set the spatial mapping matrix in the DUT to direct mapping. It causes a one-to-one mapping of space time streams to TX antennas. Thus a broken TX chain (no power) is detected and a damaged chain is identified by its bad EVM.

Suppressed linked return values: reliability

return
power_sts_tx: Four values, one value per space-time stream

read() → List[float]

```
# SCPI: READ:WLAN:MEASurement<instance>
↪:MEValuation:MODulation:CMIMo:PSTS:MAXimum
value: List[float] = driver.wlanMeas.multiEval.modulation.cmimo.psts.maximum.
↪read()
```

Return the single value RMS power results for the individual space-time streams. The current, average, minimum, maximum, and standard deviation results can be retrieved. For a meaningful result, set the spatial mapping matrix in the DUT to direct mapping. It causes a one-to-one mapping of space time streams to TX antennas. Thus a broken TX chain (no power) is detected and a damaged chain is identified by its bad EVM.

Suppressed linked return values: reliability

return
power_sts_tx: Four values, one value per space-time stream

6.4.1.2.4.8 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<instance>:MEValuation:MODulation:CMIMo:PSTS:MINimum
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:CMIMo:PSTS:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<instance>
↪:MEValuation:MODulation:CMIMo:PSTS:MINimum
value: List[float] = driver.wlanMeas.multiEval.modulation.cmimo.psts.minimum.
↪fetch()
```

Return the single value RMS power results for the individual space-time streams. The current, average, minimum, maximum, and standard deviation results can be retrieved. For a meaningful result, set the spatial mapping matrix in the DUT to direct mapping. It causes a one-to-one mapping of space time streams to TX antennas. Thus a broken TX chain (no power) is detected and a damaged chain is identified by its bad EVM.

Suppressed linked return values: reliability

return
power_sts_tx: Four values, one value per space-time stream

read() → List[float]

```
# SCPI: READ:WLAN:MEASurement<instance>
↪:MEValuation:MODulation:CMIMo:PSTS:MINimum
value: List[float] = driver.wlanMeas.multiEval.modulation.cmimo.psts.minimum.
↪read()
```

Return the single value RMS power results for the individual space-time streams. The current, average, minimum, maximum, and standard deviation results can be retrieved. For a meaningful result, set the spatial mapping matrix in the DUT to direct mapping. It causes a one-to-one mapping of space time streams to

TX antennas. Thus a broken TX chain (no power) is detected and a damaged chain is identified by its bad EVM.

Suppressed linked return values: reliability

return
power_sts_tx: Four values, one value per space-time stream

6.4.1.2.4.9 StandardDev

SCPI Commands :

```
READ:WLAN:MEASurement<instance>:MEValuation:MODulation:CMIMo:PSTS:SDEViation
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:CMIMo:PSTS:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<instance>
↪:MEValuation:MODulation:CMIMo:PSTS:SDEViation
value: List[float] = driver.wlanMeas.multiEval.modulation.cmimo.psts.
↪standardDev.fetch()
```

Return the single value RMS power results for the individual space-time streams. The current, average, minimum, maximum, and standard deviation results can be retrieved. For a meaningful result, set the spatial mapping matrix in the DUT to direct mapping. It causes a one-to-one mapping of space time streams to TX antennas. Thus a broken TX chain (no power) is detected and a damaged chain is identified by its bad EVM.

Suppressed linked return values: reliability

return
power_sts_tx: Four values, one value per space-time stream

read() → List[float]

```
# SCPI: READ:WLAN:MEASurement<instance>
↪:MEValuation:MODulation:CMIMo:PSTS:SDEViation
value: List[float] = driver.wlanMeas.multiEval.modulation.cmimo.psts.
↪standardDev.read()
```

Return the single value RMS power results for the individual space-time streams. The current, average, minimum, maximum, and standard deviation results can be retrieved. For a meaningful result, set the spatial mapping matrix in the DUT to direct mapping. It causes a one-to-one mapping of space time streams to TX antennas. Thus a broken TX chain (no power) is detected and a damaged chain is identified by its bad EVM.

Suppressed linked return values: reliability

return
power_sts_tx: Four values, one value per space-time stream

6.4.1.2.4.10 StandardDev

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:CMIMo:SDEViation
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CMIMo:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Mcs_Index: int: No parameter help available
- Nof_Ss: int: No parameter help available
- No_Of_Sts: int: No parameter help available
- Payload_Length: int: No parameter help available
- Evm_All_Carr: float: No parameter help available
- Evm_Data_Carr: float: No parameter help available
- Evm_Pilot_Carr: float: No parameter help available
- Power_Total: float: No parameter help available
- Power_Total_Peak: float: No parameter help available
- Power_Sts_1: float: No parameter help available
- Power_Sts_2: float: No parameter help available
- Power_Sts_3: float: No parameter help available
- Power_Sts_4: float: No parameter help available
- Freq_Error: float: No parameter help available
- Out_Of_Tol: float: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CMIMo:SDEViation
value: ResultData = driver.wlanMeas.multiEval.modulation.cmimo.standardDev.
↪fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:CMIMo:SDEViation
value: ResultData = driver.wlanMeas.multiEval.modulation.cmimo.standardDev.
↪read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.2.5 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CURRent
CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:CURRent
```

`class CurrentCls`

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

`class CalculateStruct`

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tol: enums.ResultStatus2: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Mcs_Index: enums.ResultStatus2: Modulation and coding scheme index
- Mod_Type: enums.ResultStatus2: No parameter help available
- Payload_Sym: enums.ResultStatus2: Number of OFDM symbols in the payload of the measured burst
- Measured_Sym: enums.ResultStatus2: Number of measured payload OFDM symbols
- Payload_Bytes: enums.ResultStatus2: Number of bytes in the payload of the measured burst.
- Guard_Interval: enums.ResultStatus2: SHORt, LONG: short or long guard interval (up to 802.11ac) GI08, GI16, GI32: 0.8 s, 1.6 s, and 3.2 s guard interval durations (for 802.11ax)
- Nof_Ss: enums.ResultStatus2: Number of spatial streams
- No_Of_Sts: enums.ResultStatus2: Number of space-time streams
- Burst_Rate: enums.ResultStatus2: If a modulation filter is used, the burst rate indicates the share of bursts of the selected modulation format in the bursts received. Otherwise, it returns 1.
- Power_Backoff: enums.ResultStatus2: Minimum distance of signal power to reference level since the start of the measurement.
- Burst_Power: enums.ResultStatus2: RMS power of the measured burst
- Peak_Power: enums.ResultStatus2: Peak power of the measured burst
- Crest_Factor: enums.ResultStatus2: No parameter help available
- Evm_All_Carr: enums.ResultStatus2: EVM for all carriers
- Evm_Data_Carr: enums.ResultStatus2: EVM for data carriers
- Evm_Pilot_Carr: enums.ResultStatus2: EVM for pilot carriers
- Freq_Error: enums.ResultStatus2: Center frequency error
- Clock_Error: enums.ResultStatus2: Symbol clock error
- Iq_Offset: enums.ResultStatus2: No parameter help available
- Dc_Power: enums.ResultStatus2: Power of the DC subcarriers

- Gain_Imbalance: enums.ResultStatus2: Gain imbalance cannot be calculated if the spectrum is not symmetrical, e.g. for HE_TB and HE_MU.
- Quad_Error: enums.ResultStatus2: Quadrature error cannot be calculated if the spectrum is not symmetrical, e.g. for HE_TB and HE_MU.
- Ltf_Power: enums.ResultStatus2: Power of long training fields (LTF) portion
- Data_Power: enums.ResultStatus2: Power of data portion
- Preamble_Power: enums.ResultStatus2: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Mcs_Index: int: Modulation and coding scheme index
- Mod_Type: enums.ModulationTypeD: No parameter help available
- Payload_Sym: int: Number of OFDM symbols in the payload of the measured burst
- Measured_Sym: int: Number of measured payload OFDM symbols
- Payload_Bytes: int: Number of bytes in the payload of the measured burst.
- Guard_Interval: enums.GuardInterval: SHORt, LONG: short or long guard interval (up to 802.11ac) GI08, GI16, GI32: 0.8 s, 1.6 s, and 3.2 s guard interval durations (for 802.11ax)
- Nof_Ss: int: Number of spatial streams
- No_Of_Sts: int: Number of space-time streams
- Burst_Rate: float: If a modulation filter is used, the burst rate indicates the share of bursts of the selected modulation format in the bursts received. Otherwise, it returns 1.
- Power_Backoff: float: Minimum distance of signal power to reference level since the start of the measurement.
- Burst_Power: float: RMS power of the measured burst
- Peak_Power: float: Peak power of the measured burst
- Crest_Factor: float: No parameter help available
- Evm_All_Carr: float: EVM for all carriers
- Evm_Data_Carr: float: EVM for data carriers
- Evm_Pilot_Carr: float: EVM for pilot carriers
- Freq_Error: float: Center frequency error
- Clock_Error: float: Symbol clock error
- Iq_Offset: float: No parameter help available
- Dc_Power: float: Power of the DC subcarriers
- Gain_Imbalance: float: Gain imbalance cannot be calculated if the spectrum is not symmetrical, e.g. for HE_TB and HE_MU.
- Quad_Error: float: Quadrature error cannot be calculated if the spectrum is not symmetrical, e.g. for HE_TB and HE_MU.

- Ltf_Power: float: Power of long training fields (LTF) portion
- Data_Power: float: Power of data portion
- Preamble_Power: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:CURRent
value: CalculateStruct = driver.wlanMeas.multiEval.modulation.current.
↪calculate()
```

Return the single value results for OFDM SISO measurements. For MIMO measurements, the stream/antenna-independent values are returned. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CURRent
value: ResultData = driver.wlanMeas.multiEval.modulation.current.fetch()
```

Return the single value results for OFDM SISO measurements. For MIMO measurements, the stream/antenna-independent values are returned. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:CURRent
value: ResultData = driver.wlanMeas.multiEval.modulation.current.read()
```

Return the single value results for OFDM SISO measurements. For MIMO measurements, the stream/antenna-independent values are returned. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.2.6 Dsss

class DsssCls

Dsss commands group definition. 15 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.modulation.dsss.clone()
```

Subgroups

6.4.1.2.6.1 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:DSSS:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:DSSS:AVERage
CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:DSSS:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: str: 'Reliability indicator'
- Mod_Type: enums.ResultStatus2: DBPSk1: 1 Mbit/s DBPSK DQPSk2: 2 Mbit/s DQPSK CCK5: 5.5 Mbit/s CCK CCK11: 11 Mbit/s CCK
- Plcp_Type: enums.ResultStatus2: Short or long PLCP
- Payload_Length: enums.ResultStatus2: No parameter help available
- Burst_Power: enums.ResultStatus2: No parameter help available
- Evm_Peak: enums.ResultStatus2: Error vector magnitude peak value
- Evm: enums.ResultStatus2: No parameter help available
- Freq_Error: enums.ResultStatus2: Center frequency error
- Clock_Error: enums.ResultStatus2: Chip clock error
- Iq_Offset: enums.ResultStatus2: No parameter help available
- Gain_Imbalance: enums.ResultStatus2: Gain imbalance
- Quad_Error: enums.ResultStatus2: Quadrature error
- Out_Of_Tol: enums.ResultStatus2: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Burst_Rate: enums.ResultStatus2: If a modulation filter is used, the burst rate indicates the share of bursts of the selected modulation format in the bursts received. Otherwise, it returns 1.

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Mod_Type: enums.ModulationTypeC: DBPSk1: 1 Mbit/s DBPSK DQPSk2: 2 Mbit/s DQPSK CCK5: 5.5 Mbit/s CCK CCK11: 11 Mbit/s CCK
- Plcp_Type: enums.PlcpType: Short or long PLCP

- Payload_Length: int: No parameter help available
- Burst_Power: float: No parameter help available
- Evm_Peak: float: Error vector magnitude peak value
- Evm: float: No parameter help available
- Freq_Error: float: Center frequency error
- Clock_Error: float: Chip clock error
- Iq_Offset: float: No parameter help available
- Gain_Imbalance: float: Gain imbalance
- Quad_Error: float: Quadrature error
- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Burst_Rate: float: If a modulation filter is used, the burst rate indicates the share of bursts of the selected modulation format in the bursts received. Otherwise, it returns 1.

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:DSSS:AVERage
value: CalculateStruct = driver.wlanMeas.multiEval.modulation.dsss.average.
↪calculate()
```

Return the current, average, minimum, maximum and standard deviation single value results for DSSS signals. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:DSSS:AVERage
value: ResultData = driver.wlanMeas.multiEval.modulation.dsss.average.fetch()
```

Return the current, average, minimum, maximum and standard deviation single value results for DSSS signals. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:DSSS:AVERage
value: ResultData = driver.wlanMeas.multiEval.modulation.dsss.average.read()
```

Return the current, average, minimum, maximum and standard deviation single value results for DSSS signals. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.2.6.2 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:DSSS:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:DSSS:CURRent
CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:DSSS:CURRent
```

`class CurrentCls`

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

`class CalculateStruct`

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Mod_Type: enums.ResultStatus2: DBPSk1: 1 Mbit/s DBPSK DQPSk2: 2 Mbit/s DQPSK CCK5: 5.5 Mbit/s CCK CCK11: 11 Mbit/s CCK
- Plcp_Type: enums.ResultStatus2: Short or long PLCP
- Payload_Length: enums.ResultStatus2: No parameter help available
- Burst_Power: enums.ResultStatus2: No parameter help available
- Evm_Peak: enums.ResultStatus2: Error vector magnitude peak value
- Evm: enums.ResultStatus2: No parameter help available
- Freq_Error: enums.ResultStatus2: Center frequency error
- Clock_Error: enums.ResultStatus2: Chip clock error
- Iq_Offset: enums.ResultStatus2: No parameter help available
- Gain_Imbalance: enums.ResultStatus2: Gain imbalance
- Quad_Error: enums.ResultStatus2: Quadrature error
- Out_Of_Tol: enums.ResultStatus2: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Burst_Rate: enums.ResultStatus2: If a modulation filter is used, the burst rate indicates the share of bursts of the selected modulation format in the bursts received. Otherwise, it returns 1.

`class FetchStruct`

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Mod_Type: enums.ModulationTypeC: DBPSk1: 1 Mbit/s DBPSK DQPSk2: 2 Mbit/s DQPSK CCK5: 5.5 Mbit/s CCK CCK11: 11 Mbit/s CCK
- Plcp_Type: enums.PlcpType: Short or long PLCP
- Payload_Length: float: No parameter help available
- Burst_Power: float: No parameter help available
- Evm_Peak: float: Error vector magnitude peak value
- Evm: float: No parameter help available
- Freq_Error: float: Center frequency error

- Clock_Error: float: Chip clock error
- Iq_Offset: float: No parameter help available
- Gain_Imbalance: float: Gain imbalance
- Quad_Error: float: Quadrature error
- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Burst_Rate: float: If a modulation filter is used, the burst rate indicates the share of bursts of the selected modulation format in the bursts received. Otherwise, it returns 1.

class ReadStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Mod_Type: enums.ModulationTypeC: DBPSk1: 1 Mbit/s DBPSK DQPSk2: 2 Mbit/s DQPSK CCK5: 5.5 Mbit/s CCK CCK11: 11 Mbit/s CCK
- Plcp_Type: enums.PlcpType: Short or long PLCP
- Payload_Length: int: No parameter help available
- Burst_Power: float: No parameter help available
- Evm_Peak: float: Error vector magnitude peak value
- Evm: float: No parameter help available
- Freq_Error: float: Center frequency error
- Clock_Error: float: Chip clock error
- Iq_Offset: float: No parameter help available
- Gain_Imbalance: float: Gain imbalance
- Quad_Error: float: Quadrature error
- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Burst_Rate: float: If a modulation filter is used, the burst rate indicates the share of bursts of the selected modulation format in the bursts received. Otherwise, it returns 1.

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:DSSS:CURRent
value: CalculateStruct = driver.wlanMeas.multiEval.modulation.dsss.current.
↪calculate()
```

Return the current, average, minimum, maximum and standard deviation single value results for DSSS signals. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:DSSS:CURRent
value: FetchStruct = driver.wlanMeas.multiEval.modulation.dsss.current.fetch()
```

Return the current, average, minimum, maximum and standard deviation single value results for DSSS signals. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for FetchStruct structure arguments.

read() → ReadStruct

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:DSSS:CURRent
value: ReadStruct = driver.wlanMeas.multiEval.modulation.dsss.current.read()
```

Return the current, average, minimum, maximum and standard deviation single value results for DSSS signals. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ReadStruct structure arguments.

6.4.1.2.6.3 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:DSSS:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:DSSS:MAXimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:DSSS:MAXimum
```

class MaximumCls

Maximum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Mod_Type: enums.ResultStatus2: DBPSk1: 1 Mbit/s DBPSK DQPSk2: 2 Mbit/s DQPSK CCK5: 5.5 Mbit/s CCK CCK11: 11 Mbit/s CCK
- Plcp_Type: enums.ResultStatus2: Short or long PLCP
- Payload_Length: enums.ResultStatus2: No parameter help available
- Burst_Power: enums.ResultStatus2: No parameter help available
- Evm_Peak: enums.ResultStatus2: Error vector magnitude peak value
- Evm: enums.ResultStatus2: No parameter help available
- Freq_Error: enums.ResultStatus2: Center frequency error
- Clock_Error: enums.ResultStatus2: Chip clock error
- Iq_Offset: enums.ResultStatus2: No parameter help available
- Gain_Imbalance: enums.ResultStatus2: Gain imbalance
- Quad_Error: enums.ResultStatus2: Quadrature error
- Out_Of_Tol: enums.ResultStatus2: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.

- Burst_Rate: enums.ResultStatus2: If a modulation filter is used, the burst rate indicates the share of bursts of the selected modulation format in the bursts received. Otherwise, it returns 1.

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Mod_Type: enums.ModulationTypeC: DBPSk1: 1 Mbit/s DBPSK DQPSk2: 2 Mbit/s DQPSK CCK5: 5.5 Mbit/s CCK CCK11: 11 Mbit/s CCK
- Plcp_Type: enums.PlcpType: Short or long PLCP
- Payload_Length: int: No parameter help available
- Burst_Power: float: No parameter help available
- Evm_Peak: float: Error vector magnitude peak value
- Evm: float: No parameter help available
- Freq_Error: float: Center frequency error
- Clock_Error: float: Chip clock error
- Iq_Offset: float: No parameter help available
- Gain_Imbalance: float: Gain imbalance
- Quad_Error: float: Quadrature error
- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Burst_Rate: float: If a modulation filter is used, the burst rate indicates the share of bursts of the selected modulation format in the bursts received. Otherwise, it returns 1.

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:DSSS:MAXimum
value: CalculateStruct = driver.wlanMeas.multiEval.modulation.dsss.maximum.
↪calculate()
```

Return the current, average, minimum, maximum and standard deviation single value results for DSSS signals. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:DSSS:MAXimum
value: ResultData = driver.wlanMeas.multiEval.modulation.dsss.maximum.fetch()
```

Return the current, average, minimum, maximum and standard deviation single value results for DSSS signals. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:DSSS:MAXimum
value: ResultData = driver.wlanMeas.multiEval.modulation.dsss.maximum.read()
```

Return the current, average, minimum, maximum and standard deviation single value results for DSSS signals. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.2.6.4 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:DSSS:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:DSSS:MINimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:DSSS:MINimum
```

class MinimumCls

Minimum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Mod_Type: enums.ResultStatus2: DBPSk1: 1 Mbit/s DBPSK DQPSk2: 2 Mbit/s DQPSK CCK5: 5.5 Mbit/s CCK CCK11: 11 Mbit/s CCK
- Plcp_Type: enums.ResultStatus2: Short or long PLCP
- Payload_Length: enums.ResultStatus2: No parameter help available
- Burst_Power: enums.ResultStatus2: No parameter help available
- Evm_Peak: enums.ResultStatus2: Error vector magnitude peak value
- Evm: enums.ResultStatus2: No parameter help available
- Freq_Error: enums.ResultStatus2: Center frequency error
- Clock_Error: enums.ResultStatus2: Chip clock error
- Iq_Offset: enums.ResultStatus2: No parameter help available
- Gain_Imbalance: enums.ResultStatus2: Gain imbalance
- Quad_Error: enums.ResultStatus2: Quadrature error
- Out_Of_Tol: enums.ResultStatus2: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Burst_Rate: enums.ResultStatus2: If a modulation filter is used, the burst rate indicates the share of bursts of the selected modulation format in the bursts received. Otherwise, it returns 1.

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'

- Mod_Type: enums.ModulationTypeC: DBPSk1: 1 Mbit/s DBPSK DQPSk2: 2 Mbit/s DQPSK CCK5: 5.5 Mbit/s CCK CCK11: 11 Mbit/s CCK
- Plcp_Type: enums.PlcpType: Short or long PLCP
- Payload_Length: int: No parameter help available
- Burst_Power: float: No parameter help available
- Evm_Peak: float: Error vector magnitude peak value
- Evm: float: No parameter help available
- Freq_Error: float: Center frequency error
- Clock_Error: float: Chip clock error
- Iq_Offset: float: No parameter help available
- Gain_Imbalance: float: Gain imbalance
- Quad_Error: float: Quadrature error
- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Burst_Rate: float: If a modulation filter is used, the burst rate indicates the share of bursts of the selected modulation format in the bursts received. Otherwise, it returns 1.

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:DSSS:MINimum
value: CalculateStruct = driver.wlanMeas.multiEval.modulation.dsss.minimum.
↪calculate()
```

Return the current, average, minimum, maximum and standard deviation single value results for DSSS signals. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:DSSS:MINimum
value: ResultData = driver.wlanMeas.multiEval.modulation.dsss.minimum.fetch()
```

Return the current, average, minimum, maximum and standard deviation single value results for DSSS signals. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:DSSS:MINimum
value: ResultData = driver.wlanMeas.multiEval.modulation.dsss.minimum.read()
```

Return the current, average, minimum, maximum and standard deviation single value results for DSSS signals. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.2.6.5 StandardDev

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:DSSS:SDEViation
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:DSSS:SDEViation
CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:DSSS:SDEViation
```

`class StandardDevCls`

StandardDev commands group definition. 3 total commands, 0 Subgroups, 3 group commands

`class CalculateStruct`

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Mod_Type: enums.ResultStatus2: DBPSk1: 1 Mbit/s DBPSK DQPSk2: 2 Mbit/s DQPSK CCK5: 5.5 Mbit/s CCK CCK11: 11 Mbit/s CCK
- Plcp_Type: enums.ResultStatus2: Short or long PLCP
- Payload_Length: enums.ResultStatus2: No parameter help available
- Burst_Power: enums.ResultStatus2: No parameter help available
- Evm_Peak: enums.ResultStatus2: Error vector magnitude peak value
- Evm: enums.ResultStatus2: No parameter help available
- Freq_Error: enums.ResultStatus2: Center frequency error
- Clock_Error: enums.ResultStatus2: Chip clock error
- Iq_Offset: enums.ResultStatus2: No parameter help available
- Gain_Imbalance: enums.ResultStatus2: Gain imbalance
- Quad_Error: enums.ResultStatus2: Quadrature error
- Out_Of_Tol: enums.ResultStatus2: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Burst_Rate: enums.ResultStatus2: If a modulation filter is used, the burst rate indicates the share of bursts of the selected modulation format in the bursts received. Otherwise, it returns 1.

`class ResultData`

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Mod_Type: enums.ModulationTypeC: DBPSk1: 1 Mbit/s DBPSK DQPSk2: 2 Mbit/s DQPSK CCK5: 5.5 Mbit/s CCK CCK11: 11 Mbit/s CCK
- Plcp_Type: enums.PlcpType: Short or long PLCP
- Payload_Length: int: No parameter help available
- Burst_Power: float: No parameter help available
- Evm_Peak: float: Error vector magnitude peak value

- Evm: float: No parameter help available
- Freq_Error: float: Center frequency error
- Clock_Error: float: Chip clock error
- Iq_Offset: float: No parameter help available
- Gain_Imbalance: float: Gain imbalance
- Quad_Error: float: Quadrature error
- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Burst_Rate: float: If a modulation filter is used, the burst rate indicates the share of bursts of the selected modulation format in the bursts received. Otherwise, it returns 1.

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>
↪:MEValuation:MODulation:DSSS:SDEViation
value: CalculateStruct = driver.wlanMeas.multiEval.modulation.dsss.standardDev.
↪calculate()
```

Return the current, average, minimum, maximum and standard deviation single value results for DSSS signals. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:DSSS:SDEViation
value: ResultData = driver.wlanMeas.multiEval.modulation.dsss.standardDev.
↪fetch()
```

Return the current, average, minimum, maximum and standard deviation single value results for DSSS signals. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:DSSS:SDEViation
value: ResultData = driver.wlanMeas.multiEval.modulation.dsss.standardDev.read()
```

Return the current, average, minimum, maximum and standard deviation single value results for DSSS signals. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.2.7 EvMagnitude

class EvMagnitudeCls

EvMagnitude commands group definition. 12 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.modulation.evMagnitude.clone()
```

Subgroups

6.4.1.2.7.1 Average

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:EVMagnitude:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability indicator'
- Evm_All_Users_All: List[float]: No parameter help available
- Evm_All_Users_Data: List[float]: No parameter help available
- Evm_All_Users_Pilot: List[float]: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>
↪:MEValuation:MODulation:EVMagnitude:AVERage
value: FetchStruct = driver.wlanMeas.multiEval.modulation.evMagnitude.average.
↪fetch()
```

Return the single value results per user for OFDMA SISO measurements. For MIMO measurements, the stream/antenna-independent values are returned. There are current, average, minimum, maximum and standard deviation results.

return

structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.2.7.2 Current

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:EVMagnitude:CURRent
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability indicator'
- Evm_All_Users_All: List[float]: No parameter help available
- Evm_All_Users_Data: List[float]: No parameter help available
- Evm_All_Users_Pilot: List[float]: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>
↪:MEValuation:MODulation:EVMagnitude:CURRent
value: FetchStruct = driver.wlanMeas.multiEval.modulation.evMagnitude.current.
↪fetch()
```

Return the single value results per user for OFDMA SISO measurements. For MIMO measurements, the stream/antenna-independent values are returned. There are current, average, minimum, maximum and standard deviation results.

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.2.7.3 Maximum

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:EVMagnitude:MAXimum
```

class MaximumCls

Maximum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability indicator'
- Evm_All_Users_All: List[float]: No parameter help available
- Evm_All_Users_Data: List[float]: No parameter help available
- Evm_All_Users_Pilot: List[float]: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>
↪:MEValuation:MODulation:EVMagnitude:MAXimum
value: FetchStruct = driver.wlanMeas.multiEval.modulation.evMagnitude.maximum.
↪fetch()
```

Return the single value results per user for OFDMA SISO measurements. For MIMO measurements, the stream/antenna-independent values are returned. There are current, average, minimum, maximum and standard deviation results.

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.2.7.4 StandardDev

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:EVMagnitude:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability indicator'
- Evm_All_Users_All: List[float]: No parameter help available
- Evm_All_Users_Data: List[float]: No parameter help available
- Evm_All_Users_Pilot: List[float]: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>
↪:MEValuation:MODulation:EVMagnitude:SDEViation
value: FetchStruct = driver.wlanMeas.multiEval.modulation.evMagnitude.
↪standardDev.fetch()
```

Return the single value results per user for OFDMA SISO measurements. For MIMO measurements, the stream/antenna-independent values are returned. There are current, average, minimum, maximum and standard deviation results.

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.2.7.5 User<User>

RepCap Settings

```
# Range: Nr1 .. Nr144
rc = driver.wlanMeas.multiEval.modulation.evMagnitude.user.repcap_user_get()
driver.wlanMeas.multiEval.modulation.evMagnitude.user.repcap_user_set(repcap.User.Nr1)
```

class UserCls

User commands group definition. 8 total commands, 5 Subgroups, 0 group commands Repeated Capability: User, default value after init: User.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.modulation.evMagnitude.user.clone()
```

Subgroups

6.4.1.2.7.6 Average

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:EVMagnitude:USER<user>:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability indicator'
- Evm_Vs_User_All: float: No parameter help available
- Evm_Vs_User_Data: float: No parameter help available
- Evm_Vs_User_Pilot: float: No parameter help available

fetch(*user=User.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:EVMagnitude:USER
↪<user>:AVERage
value: FetchStruct = driver.wlanMeas.multiEval.modulation.evMagnitude.user.
↪average.fetch(user = repcap.User.Default)
```

Return the single value results for OFDMA SISO measurements for the specified user. For MIMO measurements, the stream/antenna-independent values are returned. There are current, average, minimum, maximum and standard deviation results.

param user
optional repeated capability selector. Default value: Nr1 (settable in the interface 'User')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.2.7.7 Current

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:EVMagnitude:USER<user>:CURRent
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability indicator'
- Evm_Vs_User_All: float: No parameter help available
- Evm_Vs_User_Data: float: No parameter help available
- Evm_Vs_User_Pilot: float: No parameter help available

fetch(*user=User.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:EVMagnitude:USER
↪<user>:CURRent
value: FetchStruct = driver.wlanMeas.multiEval.modulation.evMagnitude.user.
↪current.fetch(user = repcap.User.Default)
```

Return the single value results for OFDMA SISO measurements for the specified user. For MIMO measurements, the stream/antenna-independent values are returned. There are current, average, minimum, maximum and standard deviation results.

param user
optional repeated capability selector. Default value: Nr1 (settable in the interface 'User')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.2.7.8 Maximum

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:EVMagnitude:USER<user>:MAXimum
```

class MaximumCls

Maximum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability indicator'
- Evm_Vs_User_All: float: No parameter help available

- Evm_Vs_User_Data: float: No parameter help available
- Evm_Vs_User_Pilot: float: No parameter help available

fetch(*user=User.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:EVMagnitude:USER
↪<user>:MAXimum
value: FetchStruct = driver.wlanMeas.multiEval.modulation.evMagnitude.user.
↪maximum.fetch(user = repcap.User.Default)
```

Return the single value results for OFDMA SISO measurements for the specified user. For MIMO measurements, the stream/antenna-independent values are returned. There are current, average, minimum, maximum and standard deviation results.

param user
optional repeated capability selector. Default value: Nr1 (settable in the interface 'User')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.2.7.9 StandardDev

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:EVMagnitude:USER<user>:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability indicator'
- Evm_Vs_User_All: float: No parameter help available
- Evm_Vs_User_Data: float: No parameter help available
- Evm_Vs_User_Pilot: float: No parameter help available

fetch(*user=User.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:EVMagnitude:USER
↪<user>:SDEViation
value: FetchStruct = driver.wlanMeas.multiEval.modulation.evMagnitude.user.
↪standardDev.fetch(user = repcap.User.Default)
```

Return the single value results for OFDMA SISO measurements for the specified user. For MIMO measurements, the stream/antenna-independent values are returned. There are current, average, minimum, maximum and standard deviation results.

param user
optional repeated capability selector. Default value: Nr1 (settable in the interface 'User')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.2.7.10 Stream<Stream>

RepCap Settings

```
# Range: Nr1 .. Nr8
rc = driver.wlanMeas.multiEval.modulation.evMagnitude.user.stream.repcap_stream_get()
driver.wlanMeas.multiEval.modulation.evMagnitude.user.stream.repcap_stream_set(repcap.
↪Stream.Nr1)
```

class StreamCls

Stream commands group definition. 4 total commands, 4 Subgroups, 0 group commands Repeated Capability: Stream, default value after init: Stream.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.modulation.evMagnitude.user.stream.clone()
```

Subgroups

6.4.1.2.7.11 Average

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:EVMagnitude:USER<user>:STReam
↪<str>:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability indicator'
- Evm_Vs_Stream_Vs_User_All: float: No parameter help available
- Evm_Vs_Stream_Vs_User_Data: float: No parameter help available
- Evm_Vs_Stream_Vs_User_Pilot: float: No parameter help available

fetch(*user=User.Default*, *stream=Stream.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:EVMagnitude:USER
↪<user>:STReam<str>:AVERage
value: FetchStruct = driver.wlanMeas.multiEval.modulation.evMagnitude.user.
↪stream.average.fetch(user = repcap.User.Default, stream = repcap.Stream.
↪Default)
```

Return the single value results for OFDMA MIMO measurements for the specified user and stream. There are current, average, minimum, maximum and standard deviation results.

param user
optional repeated capability selector. Default value: Nr1 (settable in the interface 'User')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.2.7.12 Current

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:EVMagnitude:USER<user>:STReam
↪<str>:CURRent
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability indicator'
- Evm_Vs_Stream_Vs_User_All: float: No parameter help available
- Evm_Vs_Stream_Vs_User_Data: float: No parameter help available
- Evm_Vs_Stream_Vs_User_Pilot: float: No parameter help available

fetch(*user=User.Default*, *stream=Stream.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:EVMagnitude:USER
↪<user>:STReam<str>:CURRent
value: FetchStruct = driver.wlanMeas.multiEval.modulation.evMagnitude.user.
↪stream.current.fetch(user = repcap.User.Default, stream = repcap.Stream.
↪Default)
```

Return the single value results for OFDMA MIMO measurements for the specified user and stream. There are current, average, minimum, maximum and standard deviation results.

param user
optional repeated capability selector. Default value: Nr1 (settable in the interface 'User')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.2.7.13 Maximum

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:EVMagnitude:USER<user>:STReam
↪<str>:MAXimum
```

class MaximumCls

Maximum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability indicator'
- Evm_Vs_Stream_Vs_User_All: float: No parameter help available
- Evm_Vs_Stream_Vs_User_Data: float: No parameter help available
- Evm_Vs_Stream_Vs_User_Pilot: float: No parameter help available

fetch(*user=User.Default*, *stream=Stream.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:EVMagnitude:USER
↪<user>:STReam<str>:MAXimum
value: FetchStruct = driver.wlanMeas.multiEval.modulation.evMagnitude.user.
↪stream.maximum.fetch(user = repcap.User.Default, stream = repcap.Stream.
↪Default)
```

Return the single value results for OFDMA MIMO measurements for the specified user and stream. There are current, average, minimum, maximum and standard deviation results.

param user
optional repeated capability selector. Default value: Nr1 (settable in the interface 'User')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.2.7.14 StandardDev

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:EVMagnitude:USER<user>:STReam
↪<str>:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability indicator'

- Evm_Vs_Stream_Vs_User_All: float: No parameter help available
- Evm_Vs_Stream_Vs_User_Data: float: No parameter help available
- Evm_Vs_Stream_Vs_User_Pilot: float: No parameter help available

fetch(*user=User.Default*, *stream=Stream.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:EVMagnitude:USER
↪<user>:STReam<str>:SDEViation
value: FetchStruct = driver.wlanMeas.multiEval.modulation.evMagnitude.user.
↪stream.standardDev.fetch(user = repcap.User.Default, stream = repcap.Stream.
↪Default)
```

Return the single value results for OFDMA MIMO measurements for the specified user and stream. There are current, average, minimum, maximum and standard deviation results.

param user
optional repeated capability selector. Default value: Nr1 (settable in the interface 'User')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.2.8 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:MAXimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:MAXimum
```

class MaximumCls

Maximum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tol: enums.ResultStatus2: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Mcs_Index: enums.ResultStatus2: Modulation and coding scheme index
- Mod_Type: enums.ResultStatus2: No parameter help available
- Payload_Sym: enums.ResultStatus2: Number of OFDM symbols in the payload of the measured burst
- Measured_Sym: enums.ResultStatus2: Number of measured payload OFDM symbols
- Payload_Bytes: enums.ResultStatus2: Number of bytes in the payload of the measured burst.
- Guard_Interval: enums.ResultStatus2: SHORt, LONG: short or long guard interval (up to 802.11ac) GI08, GI16, GI32: 0.8 s, 1.6 s, and 3.2 s guard interval durations (for 802.11ax)

- Nof_Ss: enums.ResultStatus2: Number of spatial streams
- No_Of_Sts: enums.ResultStatus2: Number of space-time streams
- Burst_Rate: enums.ResultStatus2: If a modulation filter is used, the burst rate indicates the share of bursts of the selected modulation format in the bursts received. Otherwise, it returns 1.
- Power_Backoff: enums.ResultStatus2: Minimum distance of signal power to reference level since the start of the measurement.
- Burst_Power: enums.ResultStatus2: RMS power of the measured burst
- Peak_Power: enums.ResultStatus2: Peak power of the measured burst
- Crest_Factor: enums.ResultStatus2: No parameter help available
- Evm_All_Carr: enums.ResultStatus2: EVM for all carriers
- Evm_Data_Carr: enums.ResultStatus2: EVM for data carriers
- Evm_Pilot_Carr: enums.ResultStatus2: EVM for pilot carriers
- Freq_Error: enums.ResultStatus2: Center frequency error
- Clock_Error: enums.ResultStatus2: Symbol clock error
- Iq_Offset: enums.ResultStatus2: No parameter help available
- Dc_Power: enums.ResultStatus2: Power of the DC subcarriers
- Gain_Imbalance: enums.ResultStatus2: Gain imbalance cannot be calculated if the spectrum is not symmetrical, e.g. for HE_TB and HE_MU.
- Quad_Error: enums.ResultStatus2: Quadrature error cannot be calculated if the spectrum is not symmetrical, e.g. for HE_TB and HE_MU.
- Ltf_Power: enums.ResultStatus2: Power of long training fields (LTF) portion
- Data_Power: enums.ResultStatus2: Power of data portion
- Preamble_Power: enums.ResultStatus2: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Mcs_Index: int: Modulation and coding scheme index
- Mod_Type: enums.ModulationTypeD: No parameter help available
- Payload_Sym: int: Number of OFDM symbols in the payload of the measured burst
- Measured_Sym: int: Number of measured payload OFDM symbols
- Payload_Bytes: int: Number of bytes in the payload of the measured burst.
- Guard_Interval: enums.GuardInterval: SHORt, LONG: short or long guard interval (up to 802.11ac) GI08, GI16, GI32: 0.8 s, 1.6 s, and 3.2 s guard interval durations (for 802.11ax)
- Nof_Ss: int: Number of spatial streams
- No_Of_Sts: int: Number of space-time streams
- Burst_Rate: float: If a modulation filter is used, the burst rate indicates the share of bursts of the selected modulation format in the bursts received. Otherwise, it returns 1.

- Power_Backoff: float: Minimum distance of signal power to reference level since the start of the measurement.
- Burst_Power: float: RMS power of the measured burst
- Peak_Power: float: Peak power of the measured burst
- Crest_Factor: float: No parameter help available
- Evm_All_Carr: float: EVM for all carriers
- Evm_Data_Carr: float: EVM for data carriers
- Evm_Pilot_Carr: float: EVM for pilot carriers
- Freq_Error: float: Center frequency error
- Clock_Error: float: Symbol clock error
- Iq_Offset: float: No parameter help available
- Dc_Power: float: Power of the DC subcarriers
- Gain_Imbalance: float: Gain imbalance cannot be calculated if the spectrum is not symmetrical, e.g. for HE_TB and HE_MU.
- Quad_Error: float: Quadrature error cannot be calculated if the spectrum is not symmetrical, e.g. for HE_TB and HE_MU.
- Ltf_Power: float: Power of long training fields (LTF) portion
- Data_Power: float: Power of data portion
- Preamble_Power: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:MAXimum
value: CalculateStruct = driver.wlanMeas.multiEval.modulation.maximum.
↪calculate()
```

Return the single value results for OFDM SISO measurements. For MIMO measurements, the stream/antenna-independent values are returned. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:MAXimum
value: ResultData = driver.wlanMeas.multiEval.modulation.maximum.fetch()
```

Return the single value results for OFDM SISO measurements. For MIMO measurements, the stream/antenna-independent values are returned. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:MAXimum
value: ResultData = driver.wlanMeas.multiEval.modulation.maximum.read()
```

Return the single value results for OFDM SISO measurements. For MIMO measurements, the stream/antenna-independent values are returned. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.2.9 Mimo<Mimo>

RepCap Settings

```
# Range: Nr1 .. Nr8
rc = driver.wlanMeas.multiEval.modulation.mimo.repcap_mimo_get()
driver.wlanMeas.multiEval.modulation.mimo.repcap_mimo_set(repcap.Mimo.Nr1)
```

class MimoCls

Mimo commands group definition. 30 total commands, 6 Subgroups, 0 group commands Repeated Capability: Mimo, default value after init: Mimo.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.modulation.mimo.clone()
```

Subgroups

6.4.1.2.9.1 Average

SCPI Commands :

```
READ:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>:AVERage
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>:AVERage
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Modulation_Tx: enums.ResultStatus2: No parameter help available
- Power_Backoff_Tx: enums.ResultStatus2: No parameter help available

- Burst_Power_Tx: enums.ResultStatus2: No parameter help available
- Peak_Power_Tx: enums.ResultStatus2: No parameter help available
- Crest_Factor_Tx: enums.ResultStatus2: Crest factor, antenna n
- Evm_All_Carr_Tx: enums.ResultStatus2: No parameter help available
- Evm_Data_Carr_Tx: enums.ResultStatus2: No parameter help available
- Evm_Pilot_Carr_Tx: enums.ResultStatus2: No parameter help available
- Iq_Offset_Tx: enums.ResultStatus2: No parameter help available
- Dc_Power_Tx: enums.ResultStatus2: No parameter help available
- Gain_Imbalance_Tx: enums.ResultStatus2: No parameter help available
- Quad_Error_Tx: enums.ResultStatus2: No parameter help available
- Ltf_Power_Tx: enums.ResultStatus2: No parameter help available
- Data_Power_Tx: enums.ResultStatus2: No parameter help available
- Preamble_Power_Tx: enums.ResultStatus2: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Modulation_Tx: enums.ModulationTypeD: No parameter help available
- Power_Backoff_Tx: float: No parameter help available
- Burst_Power_Tx: float: No parameter help available
- Peak_Power_Tx: float: No parameter help available
- Crest_Factor_Tx: float: Crest factor, antenna n
- Evm_All_Carr_Tx: float: No parameter help available
- Evm_Data_Carr_Tx: float: No parameter help available
- Evm_Pilot_Carr_Tx: float: No parameter help available
- Iq_Offset_Tx: float: No parameter help available
- Dc_Power_Tx: float: No parameter help available
- Gain_Imbalance_Tx: float: No parameter help available
- Quad_Error_Tx: float: No parameter help available
- Ltf_Power_Tx: float: No parameter help available
- Data_Power_Tx: float: No parameter help available
- Preamble_Power_Tx: float: No parameter help available

calculate(*mimo=Mimo.Default*) → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>
↪:AVERage
value: CalculateStruct = driver.wlanMeas.multiEval.modulation.mimo.average.
↪calculate(mimo = repcap.Mimo.Default)
```

Return the single value results for MIMO measurements. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch(*mimo=Mimo.Default*) → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>:AVERage
value: ResultData = driver.wlanMeas.multiEval.modulation.mimo.average.
↪fetch(mimo = repcap.Mimo.Default)
```

Return the single value results for MIMO measurements. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

read(*mimo=Mimo.Default*) → ResultData

```
# SCPI: READ:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>:AVERage
value: ResultData = driver.wlanMeas.multiEval.modulation.mimo.average.read(mimo␣
↪= repcap.Mimo.Default)
```

Return the single value results for MIMO measurements. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.2.9.2 Current

SCPI Commands :

```
READ:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>:CURRent
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>:CURRent
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>:CURRent
```

class CurrentCls

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Modulation_Tx: enums.ResultStatus2: No parameter help available
- Power_Backoff_Tx: enums.ResultStatus2: No parameter help available
- Burst_Power_Tx: enums.ResultStatus2: No parameter help available
- Peak_Power_Tx: enums.ResultStatus2: No parameter help available
- Crest_Factor_Tx: enums.ResultStatus2: Crest factor, antenna n
- Evm_All_Carr_Tx: enums.ResultStatus2: No parameter help available
- Evm_Data_Carr_Tx: enums.ResultStatus2: No parameter help available
- Evm_Pilot_Carr_Tx: enums.ResultStatus2: No parameter help available
- Iq_Offset_Tx: enums.ResultStatus2: No parameter help available
- Dc_Power_Tx: enums.ResultStatus2: No parameter help available
- Gain_Imbalance_Tx: enums.ResultStatus2: No parameter help available
- Quad_Error_Tx: enums.ResultStatus2: No parameter help available
- Ltf_Power_Tx: enums.ResultStatus2: No parameter help available
- Data_Power_Tx: enums.ResultStatus2: No parameter help available
- Preamble_Power_Tx: enums.ResultStatus2: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Modulation_Tx: enums.ModulationTypeD: No parameter help available
- Power_Backoff_Tx: float: No parameter help available
- Burst_Power_Tx: float: No parameter help available
- Peak_Power_Tx: float: No parameter help available
- Crest_Factor_Tx: float: Crest factor, antenna n
- Evm_All_Carr_Tx: float: No parameter help available
- Evm_Data_Carr_Tx: float: No parameter help available
- Evm_Pilot_Carr_Tx: float: No parameter help available
- Iq_Offset_Tx: float: No parameter help available
- Dc_Power_Tx: float: No parameter help available
- Gain_Imbalance_Tx: float: No parameter help available
- Quad_Error_Tx: float: No parameter help available
- Ltf_Power_Tx: float: No parameter help available
- Data_Power_Tx: float: No parameter help available
- Preamble_Power_Tx: float: No parameter help available

calculate(*mimo=Mimo.Default*) → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>
↪:CURRent
value: CalculateStruct = driver.wlanMeas.multiEval.modulation.mimo.current.
↪calculate(mimo = repcap.Mimo.Default)
```

Return the single value results for MIMO measurements. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch(*mimo=Mimo.Default*) → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>:CURRent
value: ResultData = driver.wlanMeas.multiEval.modulation.mimo.current.
↪fetch(mimo = repcap.Mimo.Default)
```

Return the single value results for MIMO measurements. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

read(*mimo=Mimo.Default*) → ResultData

```
# SCPI: READ:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>:CURRent
value: ResultData = driver.wlanMeas.multiEval.modulation.mimo.current.read(mimo␣
↪= repcap.Mimo.Default)
```

Return the single value results for MIMO measurements. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.2.9.3 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>:MAXimum
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>:MAXimum
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>:MAXimum
```

class MaximumCls

Maximum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Modulation_Tx: enums.ResultStatus2: No parameter help available
- Power_Backoff_Tx: enums.ResultStatus2: No parameter help available
- Burst_Power_Tx: enums.ResultStatus2: No parameter help available
- Peak_Power_Tx: enums.ResultStatus2: No parameter help available
- Crest_Factor_Tx: enums.ResultStatus2: Crest factor, antenna n
- Evm_All_Carr_Tx: enums.ResultStatus2: No parameter help available
- Evm_Data_Carr_Tx: enums.ResultStatus2: No parameter help available
- Evm_Pilot_Carr_Tx: enums.ResultStatus2: No parameter help available
- Iq_Offset_Tx: enums.ResultStatus2: No parameter help available
- Dc_Power_Tx: enums.ResultStatus2: No parameter help available
- Gain_Imbalance_Tx: enums.ResultStatus2: No parameter help available
- Quad_Error_Tx: enums.ResultStatus2: No parameter help available
- Ltf_Power_Tx: enums.ResultStatus2: No parameter help available
- Data_Power_Tx: enums.ResultStatus2: No parameter help available
- Preamble_Power_Tx: enums.ResultStatus2: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Modulation_Tx: enums.ModulationTypeD: No parameter help available
- Power_Backoff_Tx: float: No parameter help available
- Burst_Power_Tx: float: No parameter help available
- Peak_Power_Tx: float: No parameter help available
- Crest_Factor_Tx: float: Crest factor, antenna n
- Evm_All_Carr_Tx: float: No parameter help available
- Evm_Data_Carr_Tx: float: No parameter help available
- Evm_Pilot_Carr_Tx: float: No parameter help available

- Iq_Offset_Tx: float: No parameter help available
- Dc_Power_Tx: float: No parameter help available
- Gain_Imbalance_Tx: float: No parameter help available
- Quad_Error_Tx: float: No parameter help available
- Ltf_Power_Tx: float: No parameter help available
- Data_Power_Tx: float: No parameter help available
- Preamble_Power_Tx: float: No parameter help available

calculate(*mimo=Mimo.Default*) → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>
↪:MAXimum
value: CalculateStruct = driver.wlanMeas.multiEval.modulation.mimo.maximum.
↪calculate(mimo = repcap.Mimo.Default)
```

Return the single value results for MIMO measurements. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch(*mimo=Mimo.Default*) → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>:MAXimum
value: ResultData = driver.wlanMeas.multiEval.modulation.mimo.maximum.
↪fetch(mimo = repcap.Mimo.Default)
```

Return the single value results for MIMO measurements. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

read(*mimo=Mimo.Default*) → ResultData

```
# SCPI: READ:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>:MAXimum
value: ResultData = driver.wlanMeas.multiEval.modulation.mimo.maximum.read(mimo␣
↪= repcap.Mimo.Default)
```

Return the single value results for MIMO measurements. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.2.9.4 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>:MINimum
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>:MINimum
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>:MINimum
```

class MinimumCls

Minimum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Modulation_Tx: enums.ResultStatus2: No parameter help available
- Power_Backoff_Tx: enums.ResultStatus2: No parameter help available
- Burst_Power_Tx: enums.ResultStatus2: No parameter help available
- Peak_Power_Tx: enums.ResultStatus2: No parameter help available
- Crest_Factor_Tx: enums.ResultStatus2: Crest factor, antenna n
- Evm_All_Carr_Tx: enums.ResultStatus2: No parameter help available
- Evm_Data_Carr_Tx: enums.ResultStatus2: No parameter help available
- Evm_Pilot_Carr_Tx: enums.ResultStatus2: No parameter help available
- Iq_Offset_Tx: enums.ResultStatus2: No parameter help available
- Dc_Power_Tx: enums.ResultStatus2: No parameter help available
- Gain_Imbalance_Tx: enums.ResultStatus2: No parameter help available
- Quad_Error_Tx: enums.ResultStatus2: No parameter help available
- Ltf_Power_Tx: enums.ResultStatus2: No parameter help available
- Data_Power_Tx: enums.ResultStatus2: No parameter help available
- Preamble_Power_Tx: enums.ResultStatus2: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Modulation_Tx: enums.ModulationTypeD: No parameter help available
- Power_Backoff_Tx: float: No parameter help available
- Burst_Power_Tx: float: No parameter help available

- Peak_Power_Tx: float: No parameter help available
- Crest_Factor_Tx: float: Crest factor, antenna n
- Evm_All_Carr_Tx: float: No parameter help available
- Evm_Data_Carr_Tx: float: No parameter help available
- Evm_Pilot_Carr_Tx: float: No parameter help available
- Iq_Offset_Tx: float: No parameter help available
- Dc_Power_Tx: float: No parameter help available
- Gain_Imbalance_Tx: float: No parameter help available
- Quad_Error_Tx: float: No parameter help available
- Ltf_Power_Tx: float: No parameter help available
- Data_Power_Tx: float: No parameter help available
- Preamble_Power_Tx: float: No parameter help available

calculate(*mimo=Mimo.Default*) → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>
↪:MINimum
value: CalculateStruct = driver.wlanMeas.multiEval.modulation.mimo.minimum.
↪calculate(mimo = repcap.Mimo.Default)
```

Return the single value results for MIMO measurements. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch(*mimo=Mimo.Default*) → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>:MINimum
value: ResultData = driver.wlanMeas.multiEval.modulation.mimo.minimum.
↪fetch(mimo = repcap.Mimo.Default)
```

Return the single value results for MIMO measurements. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

read(*mimo=Mimo.Default*) → ResultData

```
# SCPI: READ:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>:MINimum
value: ResultData = driver.wlanMeas.multiEval.modulation.mimo.minimum.read(mimo␣
↪= repcap.Mimo.Default)
```

Return the single value results for MIMO measurements. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.2.9.5 Segments

class SegmentsCls

Segments commands group definition. 15 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.modulation.mimo.segments.clone()
```

Subgroups

6.4.1.2.9.6 Average

SCPI Commands :

```
READ:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>:SEGMents:AVERage
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>:SEGMents:AVERage
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>:SEGMents:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Evm_All_Carr_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Evm_All_Carr_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Evm_Data_Carr_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Evm_Data_Carr_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Evm_Pilot_Carr_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Evm_Pilot_Carr_Seg_2_Tx: enums.ResultStatus2: No parameter help available

- Power_Backoff_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Power_Backoff_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Burst_Power_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Burst_Power_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Peak_Power_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Peak_Power_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Crest_Factor_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Crest_Factor_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Iq_Offset_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Iq_Offset_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Dc_Power_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Dc_Power_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Ltf_Power_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Ltf_Power_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Data_Power_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Data_Power_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Preamble_Power_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Preamble_Power_Seg_2_Tx: enums.ResultStatus2: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Evm_All_Carr_Seg_1_Tx: float: No parameter help available
- Evm_All_Carr_Seg_2_Tx: float: No parameter help available
- Evm_Data_Carr_Seg_1_Tx: float: No parameter help available
- Evm_Data_Carr_Seg_2_Tx: float: No parameter help available
- Evm_Pilot_Carr_Seg_1_Tx: float: No parameter help available
- Evm_Pilot_Carr_Seg_2_Tx: float: No parameter help available
- Power_Backoff_Seg_1_Tx: float: No parameter help available
- Power_Backoff_Seg_2_Tx: float: No parameter help available
- Burst_Power_Seg_1_Tx: float: No parameter help available
- Burst_Power_Seg_2_Tx: float: No parameter help available
- Peak_Power_Seg_1_Tx: float: No parameter help available
- Peak_Power_Seg_2_Tx: float: No parameter help available
- Crest_Factor_Seg_1_Tx: float: No parameter help available
- Crest_Factor_Seg_2_Tx: float: No parameter help available
- Iq_Offset_Seg_1_Tx: float: No parameter help available

- Iq_Offset_Seg_2_Tx: float: No parameter help available
- Dc_Power_Seg_1_Tx: float: No parameter help available
- Dc_Power_Seg_2_Tx: float: No parameter help available
- Ltf_Power_Seg_1_Tx: float: No parameter help available
- Ltf_Power_Seg_2_Tx: float: No parameter help available
- Data_Power_Seg_1_Tx: float: No parameter help available
- Data_Power_Seg_2_Tx: float: No parameter help available
- Preamble_Power_Seg_1_Tx: float: No parameter help available
- Preamble_Power_Seg_2_Tx: float: No parameter help available

calculate(*mimo=Mimo.Default*) → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>
↪:SEGMents:AVERage
value: CalculateStruct = driver.wlanMeas.multiEval.modulation.mimo.segments.
↪average.calculate(mimo = repcap.Mimo.Default)
```

Return the single value results for MIMO measurements and bandwidths > 160 MHz. The results are available per antenna for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch(*mimo=Mimo.Default*) → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>
↪:SEGMents:AVERage
value: ResultData = driver.wlanMeas.multiEval.modulation.mimo.segments.average.
↪fetch(mimo = repcap.Mimo.Default)
```

Return the single value results for MIMO measurements and bandwidths > 160 MHz. The results are available per antenna for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

read(*mimo=Mimo.Default*) → ResultData

```
# SCPI: READ:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>
↪:SEGMents:AVERage
value: ResultData = driver.wlanMeas.multiEval.modulation.mimo.segments.average.
↪read(mimo = repcap.Mimo.Default)
```

Return the single value results for MIMO measurements and bandwidths > 160 MHz. The results are available per antenna for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.2.9.7 Current

SCPI Commands :

```
READ:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>:SEGMents:CURRent
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>:SEGMents:CURRent
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>:SEGMents:CURRent
```

class CurrentCls

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Evm_All_Carr_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Evm_All_Carr_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Evm_Data_Carr_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Evm_Data_Carr_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Evm_Pilot_Carr_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Evm_Pilot_Carr_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Power_Backoff_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Power_Backoff_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Burst_Power_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Burst_Power_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Peak_Power_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Peak_Power_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Crest_Factor_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Crest_Factor_Seg_2_Tx: enums.ResultStatus2: No parameter help available

- Iq_Offset_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Iq_Offset_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Dc_Power_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Dc_Power_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Ltf_Power_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Ltf_Power_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Data_Power_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Data_Power_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Preamble_Power_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Preamble_Power_Seg_2_Tx: enums.ResultStatus2: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Evm_All_Carr_Seg_1_Tx: float: No parameter help available
- Evm_All_Carr_Seg_2_Tx: float: No parameter help available
- Evm_Data_Carr_Seg_1_Tx: float: No parameter help available
- Evm_Data_Carr_Seg_2_Tx: float: No parameter help available
- Evm_Pilot_Carr_Seg_1_Tx: float: No parameter help available
- Evm_Pilot_Carr_Seg_2_Tx: float: No parameter help available
- Power_Backoff_Seg_1_Tx: float: No parameter help available
- Power_Backoff_Seg_2_Tx: float: No parameter help available
- Burst_Power_Seg_1_Tx: float: No parameter help available
- Burst_Power_Seg_2_Tx: float: No parameter help available
- Peak_Power_Seg_1_Tx: float: No parameter help available
- Peak_Power_Seg_2_Tx: float: No parameter help available
- Crest_Factor_Seg_1_Tx: float: No parameter help available
- Crest_Factor_Seg_2_Tx: float: No parameter help available
- Iq_Offset_Seg_1_Tx: float: No parameter help available
- Iq_Offset_Seg_2_Tx: float: No parameter help available
- Dc_Power_Seg_1_Tx: float: No parameter help available
- Dc_Power_Seg_2_Tx: float: No parameter help available
- Ltf_Power_Seg_1_Tx: float: No parameter help available
- Ltf_Power_Seg_2_Tx: float: No parameter help available
- Data_Power_Seg_1_Tx: float: No parameter help available
- Data_Power_Seg_2_Tx: float: No parameter help available
- Preamble_Power_Seg_1_Tx: float: No parameter help available

- Preamble_Power_Seg_2_Tx: float: No parameter help available

calculate(*mimo=Mimo.Default*) → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>
↪:SEGMents:CURRent
value: CalculateStruct = driver.wlanMeas.multiEval.modulation.mimo.segments.
↪current.calculate(mimo = repcap.Mimo.Default)
```

Return the single value results for MIMO measurements and bandwidths > 160 MHz. The results are available per antenna for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch(*mimo=Mimo.Default*) → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>
↪:SEGMents:CURRent
value: ResultData = driver.wlanMeas.multiEval.modulation.mimo.segments.current.
↪fetch(mimo = repcap.Mimo.Default)
```

Return the single value results for MIMO measurements and bandwidths > 160 MHz. The results are available per antenna for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

read(*mimo=Mimo.Default*) → ResultData

```
# SCPI: READ:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>
↪:SEGMents:CURRent
value: ResultData = driver.wlanMeas.multiEval.modulation.mimo.segments.current.
↪read(mimo = repcap.Mimo.Default)
```

Return the single value results for MIMO measurements and bandwidths > 160 MHz. The results are available per antenna for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.2.9.8 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>:SEGMents:MAXimum
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>:SEGMents:MAXimum
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>:SEGMents:MAXimum
```

class MaximumCls

Maximum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Evm_All_Carr_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Evm_All_Carr_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Evm_Data_Carr_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Evm_Data_Carr_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Evm_Pilot_Carr_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Evm_Pilot_Carr_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Power_Backoff_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Power_Backoff_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Burst_Power_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Burst_Power_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Peak_Power_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Peak_Power_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Crest_Factor_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Crest_Factor_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Iq_Offset_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Iq_Offset_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Dc_Power_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Dc_Power_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Ltf_Power_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Ltf_Power_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Data_Power_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Data_Power_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Preamble_Power_Seg_1_Tx: enums.ResultStatus2: No parameter help available

- Preamble_Power_Seg_2_Tx: enums.ResultStatus2: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Evm_All_Carr_Seg_1_Tx: float: No parameter help available
- Evm_All_Carr_Seg_2_Tx: float: No parameter help available
- Evm_Data_Carr_Seg_1_Tx: float: No parameter help available
- Evm_Data_Carr_Seg_2_Tx: float: No parameter help available
- Evm_Pilot_Carr_Seg_1_Tx: float: No parameter help available
- Evm_Pilot_Carr_Seg_2_Tx: float: No parameter help available
- Power_Backoff_Seg_1_Tx: float: No parameter help available
- Power_Backoff_Seg_2_Tx: float: No parameter help available
- Burst_Power_Seg_1_Tx: float: No parameter help available
- Burst_Power_Seg_2_Tx: float: No parameter help available
- Peak_Power_Seg_1_Tx: float: No parameter help available
- Peak_Power_Seg_2_Tx: float: No parameter help available
- Crest_Factor_Seg_1_Tx: float: No parameter help available
- Crest_Factor_Seg_2_Tx: float: No parameter help available
- Iq_Offset_Seg_1_Tx: float: No parameter help available
- Iq_Offset_Seg_2_Tx: float: No parameter help available
- Dc_Power_Seg_1_Tx: float: No parameter help available
- Dc_Power_Seg_2_Tx: float: No parameter help available
- Ltf_Power_Seg_1_Tx: float: No parameter help available
- Ltf_Power_Seg_2_Tx: float: No parameter help available
- Data_Power_Seg_1_Tx: float: No parameter help available
- Data_Power_Seg_2_Tx: float: No parameter help available
- Preamble_Power_Seg_1_Tx: float: No parameter help available
- Preamble_Power_Seg_2_Tx: float: No parameter help available

calculate(*mimo=Mimo.Default*) → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>
↪:SEGMents:MAXimum
value: CalculateStruct = driver.wlanMeas.multiEval.modulation.mimo.segments.
↪maximum.calculate(mimo = repcap.Mimo.Default)
```

Return the single value results for MIMO measurements and bandwidths > 160 MHz. The results are available per antenna for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch(*mimo=Mimo.Default*) → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>
↪:SEGMents:MAXimum
value: ResultData = driver.wlanMeas.multiEval.modulation.mimo.segments.maximum.
↪fetch(mimo = repcap.Mimo.Default)
```

Return the single value results for MIMO measurements and bandwidths > 160 MHz. The results are available per antenna for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

read(*mimo=Mimo.Default*) → ResultData

```
# SCPI: READ:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>
↪:SEGMents:MAXimum
value: ResultData = driver.wlanMeas.multiEval.modulation.mimo.segments.maximum.
↪read(mimo = repcap.Mimo.Default)
```

Return the single value results for MIMO measurements and bandwidths > 160 MHz. The results are available per antenna for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.2.9.9 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>:SEGMents:MINimum
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>:SEGMents:MINimum
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>:SEGMents:MINimum
```

class MinimumCls

Minimum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Evm_All_Carr_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Evm_All_Carr_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Evm_Data_Carr_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Evm_Data_Carr_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Evm_Pilot_Carr_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Evm_Pilot_Carr_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Power_Backoff_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Power_Backoff_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Burst_Power_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Burst_Power_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Peak_Power_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Peak_Power_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Crest_Factor_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Crest_Factor_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Iq_Offset_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Iq_Offset_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Dc_Power_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Dc_Power_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Ltf_Power_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Ltf_Power_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Data_Power_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Data_Power_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Preamble_Power_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Preamble_Power_Seg_2_Tx: enums.ResultStatus2: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Evm_All_Carr_Seg_1_Tx: float: No parameter help available
- Evm_All_Carr_Seg_2_Tx: float: No parameter help available
- Evm_Data_Carr_Seg_1_Tx: float: No parameter help available
- Evm_Data_Carr_Seg_2_Tx: float: No parameter help available
- Evm_Pilot_Carr_Seg_1_Tx: float: No parameter help available
- Evm_Pilot_Carr_Seg_2_Tx: float: No parameter help available
- Power_Backoff_Seg_1_Tx: float: No parameter help available
- Power_Backoff_Seg_2_Tx: float: No parameter help available
- Burst_Power_Seg_1_Tx: float: No parameter help available
- Burst_Power_Seg_2_Tx: float: No parameter help available
- Peak_Power_Seg_1_Tx: float: No parameter help available
- Peak_Power_Seg_2_Tx: float: No parameter help available
- Crest_Factor_Seg_1_Tx: float: No parameter help available
- Crest_Factor_Seg_2_Tx: float: No parameter help available
- Iq_Offset_Seg_1_Tx: float: No parameter help available
- Iq_Offset_Seg_2_Tx: float: No parameter help available
- Dc_Power_Seg_1_Tx: float: No parameter help available
- Dc_Power_Seg_2_Tx: float: No parameter help available
- Ltf_Power_Seg_1_Tx: float: No parameter help available
- Ltf_Power_Seg_2_Tx: float: No parameter help available
- Data_Power_Seg_1_Tx: float: No parameter help available
- Data_Power_Seg_2_Tx: float: No parameter help available
- Preamble_Power_Seg_1_Tx: float: No parameter help available
- Preamble_Power_Seg_2_Tx: float: No parameter help available

calculate(*mimo=Mimo.Default*) → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>
↪:SEGMents:MINimum
value: CalculateStruct = driver.wlanMeas.multiEval.modulation.mimo.segments.
↪minimum.calculate(mimo = repcap.Mimo.Default)
```

Return the single value results for MIMO measurements and bandwidths > 160 MHz. The results are available per antenna for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch(*mimo=Mimo.Default*) → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>
↪:SEGMents:MINimum
value: ResultData = driver.wlanMeas.multiEval.modulation.mimo.segments.minimum.
↪fetch(mimo = repcap.Mimo.Default)
```

Return the single value results for MIMO measurements and bandwidths > 160 MHz. The results are available per antenna for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

read(*mimo=Mimo.Default*) → ResultData

```
# SCPI: READ:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>
↪:SEGMents:MINimum
value: ResultData = driver.wlanMeas.multiEval.modulation.mimo.segments.minimum.
↪read(mimo = repcap.Mimo.Default)
```

Return the single value results for MIMO measurements and bandwidths > 160 MHz. The results are available per antenna for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.2.9.10 StandardDev

SCPI Commands :

```
READ:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>:SEGMents:SDEViation
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>:SEGMents:SDEViation
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>:SEGMents:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Evm_All_Carr_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Evm_All_Carr_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Evm_Data_Carr_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Evm_Data_Carr_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Evm_Pilot_Carr_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Evm_Pilot_Carr_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Power_Backoff_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Power_Backoff_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Burst_Power_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Burst_Power_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Peak_Power_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Peak_Power_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Crest_Factor_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Crest_Factor_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Iq_Offset_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Iq_Offset_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Dc_Power_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Dc_Power_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Ltf_Power_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Ltf_Power_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Data_Power_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Data_Power_Seg_2_Tx: enums.ResultStatus2: No parameter help available
- Preamble_Power_Seg_1_Tx: enums.ResultStatus2: No parameter help available
- Preamble_Power_Seg_2_Tx: enums.ResultStatus2: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Evm_All_Carr_Seg_1_Tx: float: No parameter help available
- Evm_All_Carr_Seg_2_Tx: float: No parameter help available
- Evm_Data_Carr_Seg_1_Tx: float: No parameter help available
- Evm_Data_Carr_Seg_2_Tx: float: No parameter help available
- Evm_Pilot_Carr_Seg_1_Tx: float: No parameter help available
- Evm_Pilot_Carr_Seg_2_Tx: float: No parameter help available
- Power_Backoff_Seg_1_Tx: float: No parameter help available
- Power_Backoff_Seg_2_Tx: float: No parameter help available
- Burst_Power_Seg_1_Tx: float: No parameter help available
- Burst_Power_Seg_2_Tx: float: No parameter help available
- Peak_Power_Seg_1_Tx: float: No parameter help available
- Peak_Power_Seg_2_Tx: float: No parameter help available
- Crest_Factor_Seg_1_Tx: float: No parameter help available
- Crest_Factor_Seg_2_Tx: float: No parameter help available
- Iq_Offset_Seg_1_Tx: float: No parameter help available
- Iq_Offset_Seg_2_Tx: float: No parameter help available
- Dc_Power_Seg_1_Tx: float: No parameter help available
- Dc_Power_Seg_2_Tx: float: No parameter help available
- Ltf_Power_Seg_1_Tx: float: No parameter help available
- Ltf_Power_Seg_2_Tx: float: No parameter help available
- Data_Power_Seg_1_Tx: float: No parameter help available
- Data_Power_Seg_2_Tx: float: No parameter help available
- Preamble_Power_Seg_1_Tx: float: No parameter help available
- Preamble_Power_Seg_2_Tx: float: No parameter help available

calculate(*mimo=Mimo.Default*) → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>
↪:SEGMents:SDEViation
value: CalculateStruct = driver.wlanMeas.multiEval.modulation.mimo.segments.
↪standardDev.calculate(mimo = repcap.Mimo.Default)
```

Return the single value results for MIMO measurements and bandwidths > 160 MHz. The results are available per antenna for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch(*mimo=Mimo.Default*) → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>
↪:SEGMents:SDEViation
value: ResultData = driver.wlanMeas.multiEval.modulation.mimo.segments.
↪standardDev.fetch(mimo = repcap.Mimo.Default)
```

Return the single value results for MIMO measurements and bandwidths > 160 MHz. The results are available per antenna for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

read(*mimo=Mimo.Default*) → ResultData

```
# SCPI: READ:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>
↪:SEGMents:SDEViation
value: ResultData = driver.wlanMeas.multiEval.modulation.mimo.segments.
↪standardDev.read(mimo = repcap.Mimo.Default)
```

Return the single value results for MIMO measurements and bandwidths > 160 MHz. The results are available per antenna for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.2.9.11 StandardDev

SCPI Commands :

```
READ:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>:SDEViation
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>:SDEViation
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Modulation_Tx: enums.ResultStatus2: No parameter help available
- Power_Backoff_Tx: enums.ResultStatus2: No parameter help available
- Burst_Power_Tx: enums.ResultStatus2: No parameter help available
- Peak_Power_Tx: enums.ResultStatus2: No parameter help available
- Crest_Factor_Tx: enums.ResultStatus2: Crest factor, antenna n
- Evm_All_Carr_Tx: enums.ResultStatus2: No parameter help available
- Evm_Data_Carr_Tx: enums.ResultStatus2: No parameter help available
- Evm_Pilot_Carr_Tx: enums.ResultStatus2: No parameter help available
- Iq_Offset_Tx: enums.ResultStatus2: No parameter help available
- Dc_Power_Tx: enums.ResultStatus2: No parameter help available
- Gain_Imbalance_Tx: enums.ResultStatus2: No parameter help available
- Quad_Error_Tx: enums.ResultStatus2: No parameter help available
- Ltf_Power_Tx: enums.ResultStatus2: No parameter help available
- Data_Power_Tx: enums.ResultStatus2: No parameter help available
- Preamble_Power_Tx: enums.ResultStatus2: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Modulation_Tx: enums.ModulationTypeD: No parameter help available
- Power_Backoff_Tx: float: No parameter help available
- Burst_Power_Tx: float: No parameter help available
- Peak_Power_Tx: float: No parameter help available
- Crest_Factor_Tx: float: Crest factor, antenna n
- Evm_All_Carr_Tx: float: No parameter help available
- Evm_Data_Carr_Tx: float: No parameter help available
- Evm_Pilot_Carr_Tx: float: No parameter help available

- Iq_Offset_Tx: float: No parameter help available
- Dc_Power_Tx: float: No parameter help available
- Gain_Imbalance_Tx: float: No parameter help available
- Quad_Error_Tx: float: No parameter help available
- Ltf_Power_Tx: float: No parameter help available
- Data_Power_Tx: float: No parameter help available
- Preamble_Power_Tx: float: No parameter help available

calculate(*mimo=Mimo.Default*) → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>
↪:SDEViation
value: CalculateStruct = driver.wlanMeas.multiEval.modulation.mimo.standardDev.
↪calculate(mimo = repcap.Mimo.Default)
```

Return the single value results for MIMO measurements. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch(*mimo=Mimo.Default*) → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>
↪:SDEViation
value: ResultData = driver.wlanMeas.multiEval.modulation.mimo.standardDev.
↪fetch(mimo = repcap.Mimo.Default)
```

Return the single value results for MIMO measurements. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

read(*mimo=Mimo.Default*) → ResultData

```
# SCPI: READ:WLAN:MEASurement<instance>:MEValuation:MODulation:MIMO<n>
↪:SDEViation
value: ResultData = driver.wlanMeas.multiEval.modulation.mimo.standardDev.
↪read(mimo = repcap.Mimo.Default)
```

Return the single value results for MIMO measurements. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.2.10 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:MINimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:MINimum
```

class MinimumCls

Minimum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tol: enums.ResultStatus2: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Mcs_Index: enums.ResultStatus2: Modulation and coding scheme index
- Mod_Type: enums.ResultStatus2: No parameter help available
- Payload_Sym: enums.ResultStatus2: Number of OFDM symbols in the payload of the measured burst
- Measured_Sym: enums.ResultStatus2: Number of measured payload OFDM symbols
- Payload_Bytes: enums.ResultStatus2: Number of bytes in the payload of the measured burst.
- Guard_Interval: enums.ResultStatus2: SHORt, LONG: short or long guard interval (up to 802.11ac) GI08, GI16, GI32: 0.8 s, 1.6 s, and 3.2 s guard interval durations (for 802.11ax)
- Nof_Ss: enums.ResultStatus2: Number of spatial streams
- No_Of_Sts: enums.ResultStatus2: Number of space-time streams
- Burst_Rate: enums.ResultStatus2: If a modulation filter is used, the burst rate indicates the share of bursts of the selected modulation format in the bursts received. Otherwise, it returns 1.
- Power_Backoff: enums.ResultStatus2: Minimum distance of signal power to reference level since the start of the measurement.
- Burst_Power: enums.ResultStatus2: RMS power of the measured burst
- Peak_Power: enums.ResultStatus2: Peak power of the measured burst
- Crest_Factor: enums.ResultStatus2: No parameter help available
- Evm_All_Carr: enums.ResultStatus2: EVM for all carriers
- Evm_Data_Carr: enums.ResultStatus2: EVM for data carriers
- Evm_Pilot_Carr: enums.ResultStatus2: EVM for pilot carriers
- Freq_Error: enums.ResultStatus2: Center frequency error

- Clock_Error: enums.ResultStatus2: Symbol clock error
- Iq_Offset: enums.ResultStatus2: No parameter help available
- Dc_Power: enums.ResultStatus2: Power of the DC subcarriers
- Gain_Imbalance: enums.ResultStatus2: Gain imbalance cannot be calculated if the spectrum is not symmetrical, e.g. for HE_TB and HE_MU.
- Quad_Error: enums.ResultStatus2: Quadrature error cannot be calculated if the spectrum is not symmetrical, e.g. for HE_TB and HE_MU.
- Ltf_Power: enums.ResultStatus2: Power of long training fields (LTF) portion
- Data_Power: enums.ResultStatus2: Power of data portion
- Preamble_Power: enums.ResultStatus2: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Mcs_Index: int: Modulation and coding scheme index
- Mod_Type: enums.ModulationTypeD: No parameter help available
- Payload_Sym: int: Number of OFDM symbols in the payload of the measured burst
- Measured_Sym: int: Number of measured payload OFDM symbols
- Payload_Bytes: int: Number of bytes in the payload of the measured burst.
- Guard_Interval: enums.GuardInterval: SHORt, LONG: short or long guard interval (up to 802.11ac) GI08, GI16, GI32: 0.8 s, 1.6 s, and 3.2 s guard interval durations (for 802.11ax)
- Nof_Ss: int: Number of spatial streams
- No_Of_Sts: int: Number of space-time streams
- Burst_Rate: float: If a modulation filter is used, the burst rate indicates the share of bursts of the selected modulation format in the bursts received. Otherwise, it returns 1.
- Power_Backoff: float: Minimum distance of signal power to reference level since the start of the measurement.
- Burst_Power: float: RMS power of the measured burst
- Peak_Power: float: Peak power of the measured burst
- Crest_Factor: float: No parameter help available
- Evm_All_Carr: float: EVM for all carriers
- Evm_Data_Carr: float: EVM for data carriers
- Evm_Pilot_Carr: float: EVM for pilot carriers
- Freq_Error: float: Center frequency error
- Clock_Error: float: Symbol clock error
- Iq_Offset: float: No parameter help available
- Dc_Power: float: Power of the DC subcarriers

- Gain_Imbalance: float: Gain imbalance cannot be calculated if the spectrum is not symmetrical, e.g. for HE_TB and HE_MU.
- Quad_Error: float: Quadrature error cannot be calculated if the spectrum is not symmetrical, e.g. for HE_TB and HE_MU.
- Ltf_Power: float: Power of long training fields (LTF) portion
- Data_Power: float: Power of data portion
- Preamble_Power: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:MINimum
value: CalculateStruct = driver.wlanMeas.multiEval.modulation.minimum.
↪calculate()
```

Return the single value results for OFDM SISO measurements. For MIMO measurements, the stream/antenna-independent values are returned. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:MINimum
value: ResultData = driver.wlanMeas.multiEval.modulation.minimum.fetch()
```

Return the single value results for OFDM SISO measurements. For MIMO measurements, the stream/antenna-independent values are returned. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:MINimum
value: ResultData = driver.wlanMeas.multiEval.modulation.minimum.read()
```

Return the single value results for OFDM SISO measurements. For MIMO measurements, the stream/antenna-independent values are returned. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.2.11 Ofdm

class OfdmCls

Ofdm commands group definition. 12 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.modulation.ofdm.clone()
```

Subgroups

6.4.1.2.11.1 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:OFDM:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:OFDM:AVERage
CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:OFDM:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Mod_Type: enums.ResultStatus2: No parameter help available
- Payload_Length: enums.ResultStatus2: No parameter help available
- Burst_Power: enums.ResultStatus2: No parameter help available
- Evm_All_Carr: enums.ResultStatus2: No parameter help available
- Evm_Data_Carr: enums.ResultStatus2: No parameter help available
- Evm_Pilot_Carr: enums.ResultStatus2: No parameter help available
- Freq_Error: enums.ResultStatus2: No parameter help available
- Clock_Error: enums.ResultStatus2: No parameter help available
- Iq_Offset: enums.ResultStatus2: No parameter help available
- Gain_Imbalance: enums.ResultStatus2: No parameter help available
- Quad_Error: enums.ResultStatus2: No parameter help available
- Out_Of_Tol: enums.ResultStatus2: No parameter help available
- Guard_Interval: enums.ResultStatus2: No parameter help available
- Burst_Rate: enums.ResultStatus2: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Mod_Type: enums.ModulationTypeB: No parameter help available
- Payload_Length: int: No parameter help available
- Burst_Power: float: No parameter help available
- Evm_All_Carr: float: No parameter help available
- Evm_Data_Carr: float: No parameter help available
- Evm_Pilot_Carr: float: No parameter help available
- Freq_Error: float: No parameter help available
- Clock_Error: float: No parameter help available
- Iq_Offset: float: No parameter help available
- Gain_Imbalance: float: No parameter help available
- Quad_Error: float: No parameter help available
- Out_Of_Tol: float: No parameter help available
- Guard_Interval: enums.GuardInterval: No parameter help available
- Burst_Rate: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:OFDM:AVERage
value: CalculateStruct = driver.wlanMeas.multiEval.modulation.ofdm.average.
↪calculate()
```

No command help available

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:OFDM:AVERage
value: ResultData = driver.wlanMeas.multiEval.modulation.ofdm.average.fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:OFDM:AVERage
value: ResultData = driver.wlanMeas.multiEval.modulation.ofdm.average.read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.2.11.2 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:OFDM:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:OFDM:CURRent
CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:OFDM:CURRent
```

class CurrentCls

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Mod_Type: enums.ResultStatus2: No parameter help available
- Payload_Length: enums.ResultStatus2: No parameter help available
- Burst_Power: enums.ResultStatus2: No parameter help available
- Evm_All_Carr: enums.ResultStatus2: No parameter help available
- Evm_Data_Carr: enums.ResultStatus2: No parameter help available
- Evm_Pilot_Carr: enums.ResultStatus2: No parameter help available
- Freq_Error: enums.ResultStatus2: No parameter help available
- Clock_Error: enums.ResultStatus2: No parameter help available
- Iq_Offset: enums.ResultStatus2: No parameter help available
- Gain_Imbalance: enums.ResultStatus2: No parameter help available
- Quad_Error: enums.ResultStatus2: No parameter help available
- Out_Of_Tol: enums.ResultStatus2: No parameter help available
- Guard_Interval: enums.ResultStatus2: No parameter help available
- Burst_Rate: enums.ResultStatus2: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Mod_Type: enums.ModulationTypeB: No parameter help available
- Payload_Length: int: No parameter help available
- Burst_Power: float: No parameter help available
- Evm_All_Carr: float: No parameter help available
- Evm_Data_Carr: float: No parameter help available
- Evm_Pilot_Carr: float: No parameter help available
- Freq_Error: float: No parameter help available
- Clock_Error: float: No parameter help available
- Iq_Offset: float: No parameter help available

- Gain_Imbalance: float: No parameter help available
- Quad_Error: float: No parameter help available
- Out_Of_Tol: float: No parameter help available
- Guard_Interval: enums.GuardInterval: No parameter help available
- Burst_Rate: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:OFDM:CURRent
value: CalculateStruct = driver.wlanMeas.multiEval.modulation.ofdm.current.
↪calculate()
```

No command help available

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:OFDM:CURRent
value: ResultData = driver.wlanMeas.multiEval.modulation.ofdm.current.fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:OFDM:CURRent
value: ResultData = driver.wlanMeas.multiEval.modulation.ofdm.current.read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.2.11.3 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:OFDM:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:OFDM:MAXimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:OFDM:MAXimum
```

class MaximumCls

Maximum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Mod_Type: enums.ResultStatus2: No parameter help available

- Payload_Length: enums.ResultStatus2: No parameter help available
- Burst_Power: enums.ResultStatus2: No parameter help available
- Evm_All_Carr: enums.ResultStatus2: No parameter help available
- Evm_Data_Carr: enums.ResultStatus2: No parameter help available
- Evm_Pilot_Carr: enums.ResultStatus2: No parameter help available
- Freq_Error: enums.ResultStatus2: No parameter help available
- Clock_Error: enums.ResultStatus2: No parameter help available
- Iq_Offset: enums.ResultStatus2: No parameter help available
- Gain_Imbalance: enums.ResultStatus2: No parameter help available
- Quad_Error: enums.ResultStatus2: No parameter help available
- Out_Of_Tol: enums.ResultStatus2: No parameter help available
- Guard_Interval: enums.ResultStatus2: No parameter help available
- Burst_Rate: enums.ResultStatus2: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Mod_Type: enums.ModulationTypeB: No parameter help available
- Payload_Length: int: No parameter help available
- Burst_Power: float: No parameter help available
- Evm_All_Carr: float: No parameter help available
- Evm_Data_Carr: float: No parameter help available
- Evm_Pilot_Carr: float: No parameter help available
- Freq_Error: float: No parameter help available
- Clock_Error: float: No parameter help available
- Iq_Offset: float: No parameter help available
- Gain_Imbalance: float: No parameter help available
- Quad_Error: float: No parameter help available
- Out_Of_Tol: float: No parameter help available
- Guard_Interval: enums.GuardInterval: No parameter help available
- Burst_Rate: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:OFDM:MAXimum
value: CalculateStruct = driver.wlanMeas.multiEval.modulation.ofdm.maximum.
↪calculate()
```

No command help available

return

structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:OFDM:MAXimum
value: ResultData = driver.wlanMeas.multiEval.modulation.ofdm.maximum.fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:OFDM:MAXimum
value: ResultData = driver.wlanMeas.multiEval.modulation.ofdm.maximum.read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.2.11.4 StandardDev

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:OFDM:SDEViation
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:OFDM:SDEViation
CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:OFDM:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Mod_Type: enums.ResultStatus2: No parameter help available
- Payload_Length: enums.ResultStatus2: No parameter help available
- Burst_Power: enums.ResultStatus2: No parameter help available
- Evm_All_Carr: enums.ResultStatus2: No parameter help available
- Evm_Data_Carr: enums.ResultStatus2: No parameter help available
- Evm_Pilot_Carr: enums.ResultStatus2: No parameter help available
- Freq_Error: enums.ResultStatus2: No parameter help available
- Clock_Error: enums.ResultStatus2: No parameter help available
- Iq_Offset: enums.ResultStatus2: No parameter help available
- Gain_Imbalance: enums.ResultStatus2: No parameter help available
- Quad_Error: enums.ResultStatus2: No parameter help available
- Out_Of_Tol: enums.ResultStatus2: No parameter help available
- Guard_Interval: enums.ResultStatus2: No parameter help available

- Burst_Rate: enums.ResultStatus2: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Mod_Type: enums.ModulationTypeB: No parameter help available
- Payload_Length: int: No parameter help available
- Burst_Power: float: No parameter help available
- Evm_All_Carr: float: No parameter help available
- Evm_Data_Carr: float: No parameter help available
- Evm_Pilot_Carr: float: No parameter help available
- Freq_Error: float: No parameter help available
- Clock_Error: float: No parameter help available
- Iq_Offset: float: No parameter help available
- Gain_Imbalance: float: No parameter help available
- Quad_Error: float: No parameter help available
- Out_Of_Tol: float: No parameter help available
- Guard_Interval: enums.GuardInterval: No parameter help available
- Burst_Rate: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>
↪:MEValuation:MODulation:OFDM:SDEViation
value: CalculateStruct = driver.wlanMeas.multiEval.modulation.ofdm.standardDev.
↪calculate()
```

No command help available

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:OFDM:SDEViation
value: ResultData = driver.wlanMeas.multiEval.modulation.ofdm.standardDev.
↪fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:OFDM:SDEViation
value: ResultData = driver.wlanMeas.multiEval.modulation.ofdm.standardDev.read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.2.12 Segments

class SegmentsCls

Segments commands group definition. 15 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.modulation.segments.clone()
```

Subgroups

6.4.1.2.12.1 Average

SCPI Commands :

```
READ:WLAN:MEASurement<instance>:MEValuation:MODulation:SEGMents:AVERage
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SEGMents:AVERage
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulation:SEGMents:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Evm_All_Carr_Seg_1: enums.ResultStatus2: No parameter help available
- Evm_All_Carr_Seg_2: enums.ResultStatus2: No parameter help available
- Evm_Data_Carr_Seg_1: enums.ResultStatus2: No parameter help available
- Evm_Data_Carr_Seg_2: enums.ResultStatus2: No parameter help available
- Evm_Pilot_Carr_Seg_1: enums.ResultStatus2: No parameter help available
- Evm_Pilot_Carr_Seg_2: enums.ResultStatus2: No parameter help available
- Power_Backoff_Seg_1: enums.ResultStatus2: No parameter help available
- Power_Backoff_Seg_2: enums.ResultStatus2: No parameter help available
- Burst_Power_Seg_1: enums.ResultStatus2: No parameter help available
- Burst_Power_Seg_2: enums.ResultStatus2: No parameter help available
- Peak_Power_Seg_1: enums.ResultStatus2: No parameter help available
- Peak_Power_Seg_2: enums.ResultStatus2: No parameter help available
- Crest_Factor_Seg_1: enums.ResultStatus2: No parameter help available

- Crest_Factor_Seg_2: enums.ResultStatus2: No parameter help available
- Iq_Offset_Seg_1: enums.ResultStatus2: No parameter help available
- Iq_Offset_Seg_2: enums.ResultStatus2: No parameter help available
- Dc_Power_Seg_1: enums.ResultStatus2: No parameter help available
- Dc_Power_Seg_2: enums.ResultStatus2: No parameter help available
- Ltf_Power_Seg_1: enums.ResultStatus2: No parameter help available
- Ltf_Power_Seg_2: enums.ResultStatus2: No parameter help available
- Data_Power_Seg_1: enums.ResultStatus2: No parameter help available
- Data_Power_Seg_2: enums.ResultStatus2: No parameter help available
- Preamble_Power_Seg_1: enums.ResultStatus2: No parameter help available
- Preamble_Power_Seg_2: enums.ResultStatus2: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Evm_All_Carr_Seg_1: float: No parameter help available
- Evm_All_Carr_Seg_2: float: No parameter help available
- Evm_Data_Carr_Seg_1: float: No parameter help available
- Evm_Data_Carr_Seg_2: float: No parameter help available
- Evm_Pilot_Carr_Seg_1: float: No parameter help available
- Evm_Pilot_Carr_Seg_2: float: No parameter help available
- Power_Backoff_Seg_1: float: No parameter help available
- Power_Backoff_Seg_2: float: No parameter help available
- Burst_Power_Seg_1: float: No parameter help available
- Burst_Power_Seg_2: float: No parameter help available
- Peak_Power_Seg_1: float: No parameter help available
- Peak_Power_Seg_2: float: No parameter help available
- Crest_Factor_Seg_1: float: No parameter help available
- Crest_Factor_Seg_2: float: No parameter help available
- Iq_Offset_Seg_1: float: No parameter help available
- Iq_Offset_Seg_2: float: No parameter help available
- Dc_Power_Seg_1: float: No parameter help available
- Dc_Power_Seg_2: float: No parameter help available
- Ltf_Power_Seg_1: float: No parameter help available
- Ltf_Power_Seg_2: float: No parameter help available
- Data_Power_Seg_1: float: No parameter help available
- Data_Power_Seg_2: float: No parameter help available

- Preamble_Power_Seg_1: float: No parameter help available
- Preamble_Power_Seg_2: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<instance>
↪:MEValuation:MODulation:SEGMents:AVERage
value: CalculateStruct = driver.wlanMeas.multiEval.modulation.segments.average.
↪calculate()
```

Return the single value results for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SEGMents:AVERage
value: ResultData = driver.wlanMeas.multiEval.modulation.segments.average.
↪fetch()
```

Return the single value results for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<instance>:MEValuation:MODulation:SEGMents:AVERage
value: ResultData = driver.wlanMeas.multiEval.modulation.segments.average.read()
```

Return the single value results for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.2.12.2 Current

SCPI Commands :

```
READ:WLAN:MEASurement<instance>:MEValuation:MODulation:SEGMents:CURRent
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SEGMents:CURRent
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulation:SEGMents:CURRent
```

`class CurrentCls`

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

`class CalculateStruct`

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Evm_All_Carr_Seg_1: enums.ResultStatus2: No parameter help available
- Evm_All_Carr_Seg_2: enums.ResultStatus2: No parameter help available
- Evm_Data_Carr_Seg_1: enums.ResultStatus2: No parameter help available
- Evm_Data_Carr_Seg_2: enums.ResultStatus2: No parameter help available
- Evm_Pilot_Carr_Seg_1: enums.ResultStatus2: No parameter help available
- Evm_Pilot_Carr_Seg_2: enums.ResultStatus2: No parameter help available
- Power_Backoff_Seg_1: enums.ResultStatus2: No parameter help available
- Power_Backoff_Seg_2: enums.ResultStatus2: No parameter help available
- Burst_Power_Seg_1: enums.ResultStatus2: No parameter help available
- Burst_Power_Seg_2: enums.ResultStatus2: No parameter help available
- Peak_Power_Seg_1: enums.ResultStatus2: No parameter help available
- Peak_Power_Seg_2: enums.ResultStatus2: No parameter help available
- Crest_Factor_Seg_1: enums.ResultStatus2: No parameter help available
- Crest_Factor_Seg_2: enums.ResultStatus2: No parameter help available
- Iq_Offset_Seg_1: enums.ResultStatus2: No parameter help available
- Iq_Offset_Seg_2: enums.ResultStatus2: No parameter help available
- Dc_Power_Seg_1: enums.ResultStatus2: No parameter help available
- Dc_Power_Seg_2: enums.ResultStatus2: No parameter help available
- Ltf_Power_Seg_1: enums.ResultStatus2: No parameter help available
- Ltf_Power_Seg_2: enums.ResultStatus2: No parameter help available
- Data_Power_Seg_1: enums.ResultStatus2: No parameter help available
- Data_Power_Seg_2: enums.ResultStatus2: No parameter help available
- Preamble_Power_Seg_1: enums.ResultStatus2: No parameter help available
- Preamble_Power_Seg_2: enums.ResultStatus2: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Evm_All_Carr_Seg_1: float: No parameter help available
- Evm_All_Carr_Seg_2: float: No parameter help available
- Evm_Data_Carr_Seg_1: float: No parameter help available
- Evm_Data_Carr_Seg_2: float: No parameter help available
- Evm_Pilot_Carr_Seg_1: float: No parameter help available
- Evm_Pilot_Carr_Seg_2: float: No parameter help available
- Power_Backoff_Seg_1: float: No parameter help available
- Power_Backoff_Seg_2: float: No parameter help available
- Burst_Power_Seg_1: float: No parameter help available
- Burst_Power_Seg_2: float: No parameter help available
- Peak_Power_Seg_1: float: No parameter help available
- Peak_Power_Seg_2: float: No parameter help available
- Crest_Factor_Seg_1: float: No parameter help available
- Crest_Factor_Seg_2: float: No parameter help available
- Iq_Offset_Seg_1: float: No parameter help available
- Iq_Offset_Seg_2: float: No parameter help available
- Dc_Power_Seg_1: float: No parameter help available
- Dc_Power_Seg_2: float: No parameter help available
- Ltf_Power_Seg_1: float: No parameter help available
- Ltf_Power_Seg_2: float: No parameter help available
- Data_Power_Seg_1: float: No parameter help available
- Data_Power_Seg_2: float: No parameter help available
- Preamble_Power_Seg_1: float: No parameter help available
- Preamble_Power_Seg_2: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<instance>
↪:MEValuation:MODulation:SEGMents:CURRent
value: CalculateStruct = driver.wlanMeas.multiEval.modulation.segments.current.
↪calculate()
```

Return the single value results for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SEGMents:CURRent
value: ResultData = driver.wlanMeas.multiEval.modulation.segments.current.
↪fetch()
```

Return the single value results for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<instance>:MEValuation:MODulation:SEGMents:CURRent
value: ResultData = driver.wlanMeas.multiEval.modulation.segments.current.read()
```

Return the single value results for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.2.12.3 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<instance>:MEValuation:MODulation:SEGMents:MAXimum
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SEGMents:MAXimum
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulation:SEGMents:MAXimum
```

class MaximumCls

Maximum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Evm_All_Carr_Seg_1: enums.ResultStatus2: No parameter help available
- Evm_All_Carr_Seg_2: enums.ResultStatus2: No parameter help available
- Evm_Data_Carr_Seg_1: enums.ResultStatus2: No parameter help available
- Evm_Data_Carr_Seg_2: enums.ResultStatus2: No parameter help available
- Evm_Pilot_Carr_Seg_1: enums.ResultStatus2: No parameter help available

- Evm_Pilot_Carr_Seg_2: enums.ResultStatus2: No parameter help available
- Power_Backoff_Seg_1: enums.ResultStatus2: No parameter help available
- Power_Backoff_Seg_2: enums.ResultStatus2: No parameter help available
- Burst_Power_Seg_1: enums.ResultStatus2: No parameter help available
- Burst_Power_Seg_2: enums.ResultStatus2: No parameter help available
- Peak_Power_Seg_1: enums.ResultStatus2: No parameter help available
- Peak_Power_Seg_2: enums.ResultStatus2: No parameter help available
- Crest_Factor_Seg_1: enums.ResultStatus2: No parameter help available
- Crest_Factor_Seg_2: enums.ResultStatus2: No parameter help available
- Iq_Offset_Seg_1: enums.ResultStatus2: No parameter help available
- Iq_Offset_Seg_2: enums.ResultStatus2: No parameter help available
- Dc_Power_Seg_1: enums.ResultStatus2: No parameter help available
- Dc_Power_Seg_2: enums.ResultStatus2: No parameter help available
- Ltf_Power_Seg_1: enums.ResultStatus2: No parameter help available
- Ltf_Power_Seg_2: enums.ResultStatus2: No parameter help available
- Data_Power_Seg_1: enums.ResultStatus2: No parameter help available
- Data_Power_Seg_2: enums.ResultStatus2: No parameter help available
- Preamble_Power_Seg_1: enums.ResultStatus2: No parameter help available
- Preamble_Power_Seg_2: enums.ResultStatus2: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Evm_All_Carr_Seg_1: float: No parameter help available
- Evm_All_Carr_Seg_2: float: No parameter help available
- Evm_Data_Carr_Seg_1: float: No parameter help available
- Evm_Data_Carr_Seg_2: float: No parameter help available
- Evm_Pilot_Carr_Seg_1: float: No parameter help available
- Evm_Pilot_Carr_Seg_2: float: No parameter help available
- Power_Backoff_Seg_1: float: No parameter help available
- Power_Backoff_Seg_2: float: No parameter help available
- Burst_Power_Seg_1: float: No parameter help available
- Burst_Power_Seg_2: float: No parameter help available
- Peak_Power_Seg_1: float: No parameter help available
- Peak_Power_Seg_2: float: No parameter help available
- Crest_Factor_Seg_1: float: No parameter help available
- Crest_Factor_Seg_2: float: No parameter help available

- Iq_Offset_Seg_1: float: No parameter help available
- Iq_Offset_Seg_2: float: No parameter help available
- Dc_Power_Seg_1: float: No parameter help available
- Dc_Power_Seg_2: float: No parameter help available
- Ltf_Power_Seg_1: float: No parameter help available
- Ltf_Power_Seg_2: float: No parameter help available
- Data_Power_Seg_1: float: No parameter help available
- Data_Power_Seg_2: float: No parameter help available
- Preamble_Power_Seg_1: float: No parameter help available
- Preamble_Power_Seg_2: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<instance>
↪:MEValuation:MODulation:SEGMents:MAXimum
value: CalculateStruct = driver.wlanMeas.multiEval.modulation.segments.maximum.
↪calculate()
```

Return the single value results for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SEGMents:MAXimum
value: ResultData = driver.wlanMeas.multiEval.modulation.segments.maximum.
↪fetch()
```

Return the single value results for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<instance>:MEValuation:MODulation:SEGMents:MAXimum
value: ResultData = driver.wlanMeas.multiEval.modulation.segments.maximum.read()
```

Return the single value results for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.2.12.4 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<instance>:MEValuation:MODulation:SEGMents:MINimum
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SEGMents:MINimum
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulation:SEGMents:MINimum
```

class MinimumCls

Minimum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Evm_All_Carr_Seg_1: enums.ResultStatus2: No parameter help available
- Evm_All_Carr_Seg_2: enums.ResultStatus2: No parameter help available
- Evm_Data_Carr_Seg_1: enums.ResultStatus2: No parameter help available
- Evm_Data_Carr_Seg_2: enums.ResultStatus2: No parameter help available
- Evm_Pilot_Carr_Seg_1: enums.ResultStatus2: No parameter help available
- Evm_Pilot_Carr_Seg_2: enums.ResultStatus2: No parameter help available
- Power_Backoff_Seg_1: enums.ResultStatus2: No parameter help available
- Power_Backoff_Seg_2: enums.ResultStatus2: No parameter help available
- Burst_Power_Seg_1: enums.ResultStatus2: No parameter help available
- Burst_Power_Seg_2: enums.ResultStatus2: No parameter help available
- Peak_Power_Seg_1: enums.ResultStatus2: No parameter help available
- Peak_Power_Seg_2: enums.ResultStatus2: No parameter help available
- Crest_Factor_Seg_1: enums.ResultStatus2: No parameter help available
- Crest_Factor_Seg_2: enums.ResultStatus2: No parameter help available
- Iq_Offset_Seg_1: enums.ResultStatus2: No parameter help available
- Iq_Offset_Seg_2: enums.ResultStatus2: No parameter help available
- Dc_Power_Seg_1: enums.ResultStatus2: No parameter help available
- Dc_Power_Seg_2: enums.ResultStatus2: No parameter help available
- Ltf_Power_Seg_1: enums.ResultStatus2: No parameter help available
- Ltf_Power_Seg_2: enums.ResultStatus2: No parameter help available
- Data_Power_Seg_1: enums.ResultStatus2: No parameter help available
- Data_Power_Seg_2: enums.ResultStatus2: No parameter help available
- Preamble_Power_Seg_1: enums.ResultStatus2: No parameter help available

- Preamble_Power_Seg_2: enums.ResultStatus2: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Evm_All_Carr_Seg_1: float: No parameter help available
- Evm_All_Carr_Seg_2: float: No parameter help available
- Evm_Data_Carr_Seg_1: float: No parameter help available
- Evm_Data_Carr_Seg_2: float: No parameter help available
- Evm_Pilot_Carr_Seg_1: float: No parameter help available
- Evm_Pilot_Carr_Seg_2: float: No parameter help available
- Power_Backoff_Seg_1: float: No parameter help available
- Power_Backoff_Seg_2: float: No parameter help available
- Burst_Power_Seg_1: float: No parameter help available
- Burst_Power_Seg_2: float: No parameter help available
- Peak_Power_Seg_1: float: No parameter help available
- Peak_Power_Seg_2: float: No parameter help available
- Crest_Factor_Seg_1: float: No parameter help available
- Crest_Factor_Seg_2: float: No parameter help available
- Iq_Offset_Seg_1: float: No parameter help available
- Iq_Offset_Seg_2: float: No parameter help available
- Dc_Power_Seg_1: float: No parameter help available
- Dc_Power_Seg_2: float: No parameter help available
- Ltf_Power_Seg_1: float: No parameter help available
- Ltf_Power_Seg_2: float: No parameter help available
- Data_Power_Seg_1: float: No parameter help available
- Data_Power_Seg_2: float: No parameter help available
- Preamble_Power_Seg_1: float: No parameter help available
- Preamble_Power_Seg_2: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<instance>
↪:MEValuation:MODulation:SEGMents:MINimum
value: CalculateStruct = driver.wlanMeas.multiEval.modulation.segments.minimum.
↪calculate()
```

Return the single value results for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SEGMents:MINimum
value: ResultData = driver.wlanMeas.multiEval.modulation.segments.minimum.
↪fetch()
```

Return the single value results for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<instance>:MEValuation:MODulation:SEGMents:MINimum
value: ResultData = driver.wlanMeas.multiEval.modulation.segments.minimum.read()
```

Return the single value results for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.2.12.5 StandardDev

SCPI Commands :

```
READ:WLAN:MEASurement<instance>:MEValuation:MODulation:SEGMents:SDEViation
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SEGMents:SDEViation
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulation:SEGMents:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Evm_All_Carr_Seg_1: enums.ResultStatus2: No parameter help available
- Evm_All_Carr_Seg_2: enums.ResultStatus2: No parameter help available
- Evm_Data_Carr_Seg_1: enums.ResultStatus2: No parameter help available
- Evm_Data_Carr_Seg_2: enums.ResultStatus2: No parameter help available
- Evm_Pilot_Carr_Seg_1: enums.ResultStatus2: No parameter help available

- Evm_Pilot_Carr_Seg_2: enums.ResultStatus2: No parameter help available
- Power_Backoff_Seg_1: enums.ResultStatus2: No parameter help available
- Power_Backoff_Seg_2: enums.ResultStatus2: No parameter help available
- Burst_Power_Seg_1: enums.ResultStatus2: No parameter help available
- Burst_Power_Seg_2: enums.ResultStatus2: No parameter help available
- Peak_Power_Seg_1: enums.ResultStatus2: No parameter help available
- Peak_Power_Seg_2: enums.ResultStatus2: No parameter help available
- Crest_Factor_Seg_1: enums.ResultStatus2: No parameter help available
- Crest_Factor_Seg_2: enums.ResultStatus2: No parameter help available
- Iq_Offset_Seg_1: enums.ResultStatus2: No parameter help available
- Iq_Offset_Seg_2: enums.ResultStatus2: No parameter help available
- Dc_Power_Seg_1: enums.ResultStatus2: No parameter help available
- Dc_Power_Seg_2: enums.ResultStatus2: No parameter help available
- Ltf_Power_Seg_1: enums.ResultStatus2: No parameter help available
- Ltf_Power_Seg_2: enums.ResultStatus2: No parameter help available
- Data_Power_Seg_1: enums.ResultStatus2: No parameter help available
- Data_Power_Seg_2: enums.ResultStatus2: No parameter help available
- Preamble_Power_Seg_1: enums.ResultStatus2: No parameter help available
- Preamble_Power_Seg_2: enums.ResultStatus2: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Evm_All_Carr_Seg_1: float: No parameter help available
- Evm_All_Carr_Seg_2: float: No parameter help available
- Evm_Data_Carr_Seg_1: float: No parameter help available
- Evm_Data_Carr_Seg_2: float: No parameter help available
- Evm_Pilot_Carr_Seg_1: float: No parameter help available
- Evm_Pilot_Carr_Seg_2: float: No parameter help available
- Power_Backoff_Seg_1: float: No parameter help available
- Power_Backoff_Seg_2: float: No parameter help available
- Burst_Power_Seg_1: float: No parameter help available
- Burst_Power_Seg_2: float: No parameter help available
- Peak_Power_Seg_1: float: No parameter help available
- Peak_Power_Seg_2: float: No parameter help available
- Crest_Factor_Seg_1: float: No parameter help available
- Crest_Factor_Seg_2: float: No parameter help available

- Iq_Offset_Seg_1: float: No parameter help available
- Iq_Offset_Seg_2: float: No parameter help available
- Dc_Power_Seg_1: float: No parameter help available
- Dc_Power_Seg_2: float: No parameter help available
- Ltf_Power_Seg_1: float: No parameter help available
- Ltf_Power_Seg_2: float: No parameter help available
- Data_Power_Seg_1: float: No parameter help available
- Data_Power_Seg_2: float: No parameter help available
- Preamble_Power_Seg_1: float: No parameter help available
- Preamble_Power_Seg_2: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<instance>
↪:MEValuation:MODulation:SEGMents:SDEViation
value: CalculateStruct = driver.wlanMeas.multiEval.modulation.segments.
↪standardDev.calculate()
```

Return the single value results for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<instance>
↪:MEValuation:MODulation:SEGMents:SDEViation
value: ResultData = driver.wlanMeas.multiEval.modulation.segments.standardDev.
↪fetch()
```

Return the single value results for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. There are current, average, minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<instance>
↪:MEValuation:MODulation:SEGMents:SDEViation
value: ResultData = driver.wlanMeas.multiEval.modulation.segments.standardDev.
↪read()
```

Return the single value results for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. There are current, average,

minimum, maximum and standard deviation results. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.2.13 Smimo

class SmimoCls

Smimo commands group definition. 12 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.modulation.smimo.clone()
```

Subgroups

6.4.1.2.13.1 Average

SCPI Commands :

```
READ:WLAN:MEASurement<instance>:MEValuation:MODulation:SMIMo:AVERage
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SMIMo:AVERage
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulation:SMIMo:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Mcs_Index: enums.ResultStatus2: No parameter help available
- Nof_Ss: enums.ResultStatus2: No parameter help available
- No_Of_Sts: enums.ResultStatus2: No parameter help available
- Data_Symbols: enums.ResultStatus2: No parameter help available
- Power_Total: enums.ResultStatus2: No parameter help available
- Evm_All_Carr: enums.ResultStatus2: No parameter help available
- Evm_Data_Carr: enums.ResultStatus2: No parameter help available
- Evm_Pilot_Carr: enums.ResultStatus2: No parameter help available
- Clock_Error: enums.ResultStatus2: No parameter help available
- Freq_Error: enums.ResultStatus2: No parameter help available
- Evm_All_S_1: enums.ResultStatus2: No parameter help available
- Evm_Data_S_1: enums.ResultStatus2: No parameter help available

- Evm_Pilot_S_1: enums.ResultStatus2: No parameter help available
- Evm_All_S_2: enums.ResultStatus2: No parameter help available
- Evm_Data_S_2: enums.ResultStatus2: No parameter help available
- Evm_Pilot_S_2: enums.ResultStatus2: No parameter help available
- Evm_All_S_3: enums.ResultStatus2: No parameter help available
- Evm_Data_S_3: enums.ResultStatus2: No parameter help available
- Evm_Pilot_S_3: enums.ResultStatus2: No parameter help available
- Evm_All_S_4: enums.ResultStatus2: No parameter help available
- Evm_Data_S_4: enums.ResultStatus2: No parameter help available
- Evm_Pilot_S_4: enums.ResultStatus2: No parameter help available
- Power_Tx_1: enums.ResultStatus2: No parameter help available
- Power_Tx_2: enums.ResultStatus2: No parameter help available
- Power_Tx_3: enums.ResultStatus2: No parameter help available
- Power_Tx_4: enums.ResultStatus2: No parameter help available
- Iq_Offset_1: enums.ResultStatus2: No parameter help available
- Iq_Offset_2: enums.ResultStatus2: No parameter help available
- Iq_Offset_3: enums.ResultStatus2: No parameter help available
- Iq_Offset_4: enums.ResultStatus2: No parameter help available
- Out_Of_Tol: enums.ResultStatus2: No parameter help available

class ResultData

Response structure. Fields:

- Mcs_Index: int: No parameter help available
- Nof_Ss: int: No parameter help available
- No_Of_Sts: int: No parameter help available
- Data_Symbols: int: No parameter help available
- Power_Total: float: No parameter help available
- Evm_All_Carr: float: No parameter help available
- Evm_Data_Carr: float: No parameter help available
- Evm_Pilot_Carr: float: No parameter help available
- Clock_Error: float: No parameter help available
- Freq_Error: float: No parameter help available
- Evm_All_S_1: float: No parameter help available
- Evm_Data_S_1: float: No parameter help available
- Evm_Pilot_S_1: float: No parameter help available
- Evm_All_S_2: float: No parameter help available
- Evm_Data_S_2: float: No parameter help available

- Evm_Pilot_S_2: float: No parameter help available
- Evm_All_S_3: float: No parameter help available
- Evm_Data_S_3: float: No parameter help available
- Evm_Pilot_S_3: float: No parameter help available
- Evm_All_S_4: float: No parameter help available
- Evm_Data_S_4: float: No parameter help available
- Evm_Pilot_S_4: float: No parameter help available
- Power_Tx_1: float: No parameter help available
- Power_Tx_2: float: No parameter help available
- Power_Tx_3: float: No parameter help available
- Power_Tx_4: float: No parameter help available
- Iq_Offset_1: float: No parameter help available
- Iq_Offset_2: float: No parameter help available
- Iq_Offset_3: float: No parameter help available
- Iq_Offset_4: float: No parameter help available
- Out_Of_Tol: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<instance>
↪:MEValuation:MODulation:SMIMo:AVERage
value: CalculateStruct = driver.wlanMeas.multiEval.modulation.smimo.average.
↪calculate()
```

No command help available

Suppressed linked return values: reliability

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SMIMo:AVERage
value: ResultData = driver.wlanMeas.multiEval.modulation.smimo.average.fetch()
```

No command help available

Suppressed linked return values: reliability

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<instance>:MEValuation:MODulation:SMIMo:AVERage
value: ResultData = driver.wlanMeas.multiEval.modulation.smimo.average.read()
```

No command help available

Suppressed linked return values: reliability

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.2.13.2 Current

SCPI Commands :

```
READ:WLAN:MEASurement<instance>:MEValuation:MODulation:SMIMo:CURRent
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SMIMo:CURRent
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulation:SMIMo:CURRent
```

`class CurrentCls`

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

`class CalculateStruct`

Response structure. Fields:

- Mcs_Index: enums.ResultStatus2: No parameter help available
- Nof_Ss: enums.ResultStatus2: No parameter help available
- No_Of_Sts: enums.ResultStatus2: No parameter help available
- Data_Symbols: enums.ResultStatus2: No parameter help available
- Power_Total: enums.ResultStatus2: No parameter help available
- Evm_All_Carr: enums.ResultStatus2: No parameter help available
- Evm_Data_Carr: enums.ResultStatus2: No parameter help available
- Evm_Pilot_Carr: enums.ResultStatus2: No parameter help available
- Clock_Error: enums.ResultStatus2: No parameter help available
- Freq_Error: enums.ResultStatus2: No parameter help available
- Evm_All_S_1: enums.ResultStatus2: No parameter help available
- Evm_Data_S_1: enums.ResultStatus2: No parameter help available
- Evm_Pilot_S_1: enums.ResultStatus2: No parameter help available
- Evm_All_S_2: enums.ResultStatus2: No parameter help available
- Evm_Data_S_2: enums.ResultStatus2: No parameter help available
- Evm_Pilot_S_2: enums.ResultStatus2: No parameter help available
- Evm_All_S_3: enums.ResultStatus2: No parameter help available
- Evm_Data_S_3: enums.ResultStatus2: No parameter help available
- Evm_Pilot_S_3: enums.ResultStatus2: No parameter help available
- Evm_All_S_4: enums.ResultStatus2: No parameter help available
- Evm_Data_S_4: enums.ResultStatus2: No parameter help available
- Evm_Pilot_S_4: enums.ResultStatus2: No parameter help available

- Power_Tx_1: enums.ResultStatus2: No parameter help available
- Power_Tx_2: enums.ResultStatus2: No parameter help available
- Power_Tx_3: enums.ResultStatus2: No parameter help available
- Power_Tx_4: enums.ResultStatus2: No parameter help available
- Iq_Offset_1: enums.ResultStatus2: No parameter help available
- Iq_Offset_2: enums.ResultStatus2: No parameter help available
- Iq_Offset_3: enums.ResultStatus2: No parameter help available
- Iq_Offset_4: enums.ResultStatus2: No parameter help available
- Out_Of_Tol: enums.ResultStatus2: No parameter help available

class ResultData

Response structure. Fields:

- Mcs_Index: int: No parameter help available
- Nof_Ss: int: No parameter help available
- No_Of_Sts: int: No parameter help available
- Data_Symbols: int: No parameter help available
- Power_Total: float: No parameter help available
- Evm_All_Carr: float: No parameter help available
- Evm_Data_Carr: float: No parameter help available
- Evm_Pilot_Carr: float: No parameter help available
- Clock_Error: float: No parameter help available
- Freq_Error: float: No parameter help available
- Evm_All_S_1: float: No parameter help available
- Evm_Data_S_1: float: No parameter help available
- Evm_Pilot_S_1: float: No parameter help available
- Evm_All_S_2: float: No parameter help available
- Evm_Data_S_2: float: No parameter help available
- Evm_Pilot_S_2: float: No parameter help available
- Evm_All_S_3: float: No parameter help available
- Evm_Data_S_3: float: No parameter help available
- Evm_Pilot_S_3: float: No parameter help available
- Evm_All_S_4: float: No parameter help available
- Evm_Data_S_4: float: No parameter help available
- Evm_Pilot_S_4: float: No parameter help available
- Power_Tx_1: float: No parameter help available
- Power_Tx_2: float: No parameter help available
- Power_Tx_3: float: No parameter help available

- Power_Tx_4: float: No parameter help available
- Iq_Offset_1: float: No parameter help available
- Iq_Offset_2: float: No parameter help available
- Iq_Offset_3: float: No parameter help available
- Iq_Offset_4: float: No parameter help available
- Out_Of_Tol: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<instance>
↪:MEValuation:MODulation:SMIMo:CURRent
value: CalculateStruct = driver.wlanMeas.multiEval.modulation.smimo.current.
↪calculate()
```

No command help available

Suppressed linked return values: reliability

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SMIMo:CURRent
value: ResultData = driver.wlanMeas.multiEval.modulation.smimo.current.fetch()
```

No command help available

Suppressed linked return values: reliability

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<instance>:MEValuation:MODulation:SMIMo:CURRent
value: ResultData = driver.wlanMeas.multiEval.modulation.smimo.current.read()
```

No command help available

Suppressed linked return values: reliability

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.2.13.3 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<instance>:MEValuation:MODulation:SMIMo:MAXimum
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SMIMo:MAXimum
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulation:SMIMo:MAXimum
```

class MaximumCls

Maximum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Mcs_Index: enums.ResultStatus2: No parameter help available
- Nof_Ss: enums.ResultStatus2: No parameter help available
- No_Of_Sts: enums.ResultStatus2: No parameter help available
- Data_Symbols: enums.ResultStatus2: No parameter help available
- Power_Total: enums.ResultStatus2: No parameter help available
- Evm_All_Carr: enums.ResultStatus2: No parameter help available
- Evm_Data_Carr: enums.ResultStatus2: No parameter help available
- Evm_Pilot_Carr: enums.ResultStatus2: No parameter help available
- Clock_Error: enums.ResultStatus2: No parameter help available
- Freq_Error: enums.ResultStatus2: No parameter help available
- Evm_All_S_1: enums.ResultStatus2: No parameter help available
- Evm_Data_S_1: enums.ResultStatus2: No parameter help available
- Evm_Pilot_S_1: enums.ResultStatus2: No parameter help available
- Evm_All_S_2: enums.ResultStatus2: No parameter help available
- Evm_Data_S_2: enums.ResultStatus2: No parameter help available
- Evm_Pilot_S_2: enums.ResultStatus2: No parameter help available
- Evm_All_S_3: enums.ResultStatus2: No parameter help available
- Evm_Data_S_3: enums.ResultStatus2: No parameter help available
- Evm_Pilot_S_3: enums.ResultStatus2: No parameter help available
- Evm_All_S_4: enums.ResultStatus2: No parameter help available
- Evm_Data_S_4: enums.ResultStatus2: No parameter help available
- Evm_Pilot_S_4: enums.ResultStatus2: No parameter help available
- Power_Tx_1: enums.ResultStatus2: No parameter help available
- Power_Tx_2: enums.ResultStatus2: No parameter help available
- Power_Tx_3: enums.ResultStatus2: No parameter help available
- Power_Tx_4: enums.ResultStatus2: No parameter help available
- Iq_Offset_1: enums.ResultStatus2: No parameter help available
- Iq_Offset_2: enums.ResultStatus2: No parameter help available
- Iq_Offset_3: enums.ResultStatus2: No parameter help available
- Iq_Offset_4: enums.ResultStatus2: No parameter help available
- Out_Of_Tol: enums.ResultStatus2: No parameter help available

class ResultData

Response structure. Fields:

- Mcs_Index: int: No parameter help available
- Nof_Ss: int: No parameter help available
- No_Of_Sts: int: No parameter help available
- Data_Symbols: int: No parameter help available
- Power_Total: float: No parameter help available
- Evm_All_Carr: float: No parameter help available
- Evm_Data_Carr: float: No parameter help available
- Evm_Pilot_Carr: float: No parameter help available
- Clock_Error: float: No parameter help available
- Freq_Error: float: No parameter help available
- Evm_All_S_1: float: No parameter help available
- Evm_Data_S_1: float: No parameter help available
- Evm_Pilot_S_1: float: No parameter help available
- Evm_All_S_2: float: No parameter help available
- Evm_Data_S_2: float: No parameter help available
- Evm_Pilot_S_2: float: No parameter help available
- Evm_All_S_3: float: No parameter help available
- Evm_Data_S_3: float: No parameter help available
- Evm_Pilot_S_3: float: No parameter help available
- Evm_All_S_4: float: No parameter help available
- Evm_Data_S_4: float: No parameter help available
- Evm_Pilot_S_4: float: No parameter help available
- Power_Tx_1: float: No parameter help available
- Power_Tx_2: float: No parameter help available
- Power_Tx_3: float: No parameter help available
- Power_Tx_4: float: No parameter help available
- Iq_Offset_1: float: No parameter help available
- Iq_Offset_2: float: No parameter help available
- Iq_Offset_3: float: No parameter help available
- Iq_Offset_4: float: No parameter help available
- Out_Of_Tol: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<instance>
↪:MEValuation:MODulation:SMIMo:MAXimum
value: CalculateStruct = driver.wlanMeas.multiEval.modulation.smimo.maximum.
↪calculate()
```

No command help available

Suppressed linked return values: reliability

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SMIMo:MAXimum
value: ResultData = driver.wlanMeas.multiEval.modulation.smimo.maximum.fetch()
```

No command help available

Suppressed linked return values: reliability

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<instance>:MEValuation:MODulation:SMIMo:MAXimum
value: ResultData = driver.wlanMeas.multiEval.modulation.smimo.maximum.read()
```

No command help available

Suppressed linked return values: reliability

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.2.13.4 StandardDev

SCPI Commands :

```
READ:WLAN:MEASurement<instance>:MEValuation:MODulation:SMIMo:SDEViation
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SMIMo:SDEViation
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulation:SMIMo:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Mcs_Index: enums.ResultStatus2: No parameter help available
- Nof_Ss: enums.ResultStatus2: No parameter help available
- No_Of_Sts: enums.ResultStatus2: No parameter help available
- Data_Symbols: enums.ResultStatus2: No parameter help available
- Power_Total: enums.ResultStatus2: No parameter help available

- Evm_All_Carr: enums.ResultStatus2: No parameter help available
- Evm_Data_Carr: enums.ResultStatus2: No parameter help available
- Evm_Pilot_Carr: enums.ResultStatus2: No parameter help available
- Clock_Error: enums.ResultStatus2: No parameter help available
- Freq_Error: enums.ResultStatus2: No parameter help available
- Evm_All_S_1: enums.ResultStatus2: No parameter help available
- Evm_Data_S_1: enums.ResultStatus2: No parameter help available
- Evm_Pilot_S_1: enums.ResultStatus2: No parameter help available
- Evm_All_S_2: enums.ResultStatus2: No parameter help available
- Evm_Data_S_2: enums.ResultStatus2: No parameter help available
- Evm_Pilot_S_2: enums.ResultStatus2: No parameter help available
- Evm_All_S_3: enums.ResultStatus2: No parameter help available
- Evm_Data_S_3: enums.ResultStatus2: No parameter help available
- Evm_Pilot_S_3: enums.ResultStatus2: No parameter help available
- Evm_All_S_4: enums.ResultStatus2: No parameter help available
- Evm_Data_S_4: enums.ResultStatus2: No parameter help available
- Evm_Pilot_S_4: enums.ResultStatus2: No parameter help available
- Power_Tx_1: enums.ResultStatus2: No parameter help available
- Power_Tx_2: enums.ResultStatus2: No parameter help available
- Power_Tx_3: enums.ResultStatus2: No parameter help available
- Power_Tx_4: enums.ResultStatus2: No parameter help available
- Iq_Offset_1: enums.ResultStatus2: No parameter help available
- Iq_Offset_2: enums.ResultStatus2: No parameter help available
- Iq_Offset_3: enums.ResultStatus2: No parameter help available
- Iq_Offset_4: enums.ResultStatus2: No parameter help available
- Out_Of_Tol: enums.ResultStatus2: No parameter help available

class ResultData

Response structure. Fields:

- Mcs_Index: int: No parameter help available
- Nof_Ss: int: No parameter help available
- No_Of_Sts: int: No parameter help available
- Data_Symbols: int: No parameter help available
- Power_Total: float: No parameter help available
- Evm_All_Carr: float: No parameter help available
- Evm_Data_Carr: float: No parameter help available
- Evm_Pilot_Carr: float: No parameter help available

- Clock_Error: float: No parameter help available
- Freq_Error: float: No parameter help available
- Evm_All_S_1: float: No parameter help available
- Evm_Data_S_1: float: No parameter help available
- Evm_Pilot_S_1: float: No parameter help available
- Evm_All_S_2: float: No parameter help available
- Evm_Data_S_2: float: No parameter help available
- Evm_Pilot_S_2: float: No parameter help available
- Evm_All_S_3: float: No parameter help available
- Evm_Data_S_3: float: No parameter help available
- Evm_Pilot_S_3: float: No parameter help available
- Evm_All_S_4: float: No parameter help available
- Evm_Data_S_4: float: No parameter help available
- Evm_Pilot_S_4: float: No parameter help available
- Power_Tx_1: float: No parameter help available
- Power_Tx_2: float: No parameter help available
- Power_Tx_3: float: No parameter help available
- Power_Tx_4: float: No parameter help available
- Iq_Offset_1: float: No parameter help available
- Iq_Offset_2: float: No parameter help available
- Iq_Offset_3: float: No parameter help available
- Iq_Offset_4: float: No parameter help available
- Out_Of_Tol: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<instance>
↪:MEValuation:MODulation:SMIMo:SDEViation
value: CalculateStruct = driver.wlanMeas.multiEval.modulation.smimo.standardDev.
↪calculate()
```

No command help available

Suppressed linked return values: reliability

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SMIMo:SDEViation
value: ResultData = driver.wlanMeas.multiEval.modulation.smimo.standardDev.
↪fetch()
```

No command help available

Suppressed linked return values: reliability

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<instance>:MEValuation:MODulation:SMIMo:SDEViation
value: ResultData = driver.wlanMeas.multiEval.modulation.smimo.standardDev.
↪read()
```

No command help available

Suppressed linked return values: reliability

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.2.14 StandardDev

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:SDEViation
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:SDEViation
CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tol: enums.ResultStatus2: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Mcs_Index: enums.ResultStatus2: Modulation and coding scheme index
- Mod_Type: enums.ResultStatus2: No parameter help available
- Payload_Sym: enums.ResultStatus2: Number of OFDM symbols in the payload of the measured burst
- Measured_Sym: enums.ResultStatus2: Number of measured payload OFDM symbols
- Payload_Bytes: enums.ResultStatus2: Number of bytes in the payload of the measured burst.
- Guard_Interval: enums.ResultStatus2: SHORt, LONG: short or long guard interval (up to 802.11ac) GI08, GI16, GI32: 0.8 s, 1.6 s, and 3.2 s guard interval durations (for 802.11ax)
- Nof_Ss: enums.ResultStatus2: Number of spatial streams
- No_Of_Sts: enums.ResultStatus2: Number of space-time streams
- Burst_Rate: enums.ResultStatus2: If a modulation filter is used, the burst rate indicates the share of bursts of the selected modulation format in the bursts received. Otherwise, it returns 1.
- Power_Backoff: enums.ResultStatus2: Minimum distance of signal power to reference level since the start of the measurement.

- Burst_Power: enums.ResultStatus2: RMS power of the measured burst
- Peak_Power: enums.ResultStatus2: Peak power of the measured burst
- Crest_Factor: enums.ResultStatus2: No parameter help available
- Evm_All_Carr: enums.ResultStatus2: EVM for all carriers
- Evm_Data_Carr: enums.ResultStatus2: EVM for data carriers
- Evm_Pilot_Carr: enums.ResultStatus2: EVM for pilot carriers
- Freq_Error: enums.ResultStatus2: Center frequency error
- Clock_Error: enums.ResultStatus2: Symbol clock error
- Iq_Offset: enums.ResultStatus2: No parameter help available
- Dc_Power: enums.ResultStatus2: Power of the DC subcarriers
- Gain_Imbalance: enums.ResultStatus2: Gain imbalance cannot be calculated if the spectrum is not symmetrical, e.g. for HE_TB and HE_MU.
- Quad_Error: enums.ResultStatus2: Quadrature error cannot be calculated if the spectrum is not symmetrical, e.g. for HE_TB and HE_MU.
- Ltf_Power: enums.ResultStatus2: Power of long training fields (LTF) portion
- Data_Power: enums.ResultStatus2: Power of data portion
- Preamble_Power: enums.ResultStatus2: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Mcs_Index: int: Modulation and coding scheme index
- Mod_Type: enums.ModulationTypeD: No parameter help available
- Payload_Sym: int: Number of OFDM symbols in the payload of the measured burst
- Measured_Sym: int: Number of measured payload OFDM symbols
- Payload_Bytes: int: Number of bytes in the payload of the measured burst.
- Guard_Interval: enums.GuardInterval: SHORt, LONG: short or long guard interval (up to 802.11ac) GI08, GI16, GI32: 0.8 s, 1.6 s, and 3.2 s guard interval durations (for 802.11ax)
- Nof_Ss: int: Number of spatial streams
- No_Of_Sts: int: Number of space-time streams
- Burst_Rate: float: If a modulation filter is used, the burst rate indicates the share of bursts of the selected modulation format in the bursts received. Otherwise, it returns 1.
- Power_Backoff: float: Minimum distance of signal power to reference level since the start of the measurement.
- Burst_Power: float: RMS power of the measured burst
- Peak_Power: float: Peak power of the measured burst
- Crest_Factor: float: No parameter help available

- Evm_All_Carr: float: EVM for all carriers
- Evm_Data_Carr: float: EVM for data carriers
- Evm_Pilot_Carr: float: EVM for pilot carriers
- Freq_Error: float: Center frequency error
- Clock_Error: float: Symbol clock error
- Iq_Offset: float: No parameter help available
- Dc_Power: float: Power of the DC subcarriers
- Gain_Imbalance: float: Gain imbalance cannot be calculated if the spectrum is not symmetrical, e.g. for HE_TB and HE_MU.
- Quad_Error: float: Quadrature error cannot be calculated if the spectrum is not symmetrical, e.g. for HE_TB and HE_MU.
- Ltf_Power: float: Power of long training fields (LTF) portion
- Data_Power: float: Power of data portion
- Preamble_Power: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:SDEViation
value: CalculateStruct = driver.wlanMeas.multiEval.modulation.standardDev.
↪calculate()
```

Return the single value results for OFDM SISO measurements. For MIMO measurements, the stream/antenna-independent values are returned. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:SDEViation
value: ResultData = driver.wlanMeas.multiEval.modulation.standardDev.fetch()
```

Return the single value results for OFDM SISO measurements. For MIMO measurements, the stream/antenna-independent values are returned. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:SDEViation
value: ResultData = driver.wlanMeas.multiEval.modulation.standardDev.read()
```

Return the single value results for OFDM SISO measurements. For MIMO measurements, the stream/antenna-independent values are returned. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.3 Ofdma

class OfdmaCls

Ofdma commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.ofdma.clone()
```

Subgroups

6.4.1.3.1 Info

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:OFDMa:INFO
```

class InfoCls

Info commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- No_Of_Users: int: No. of users
- No_Of_Rus: int: No. of resource units
- Guard_Interval: enums.GuardInterval: SHORt, LONG: short or long guard interval (up to 802.11ac) GI08, GI16, GI32: 0.8 s, 1.6 s, and 3.2 s guard interval durations (for 802.11ax)
- Ltf_Size: enums.LtfSize: 1x LTF (3.2 s) , 2x LTF (6.4 s) , 4x LTF (12.8 s)

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:OFDMa:INFO
value: FetchStruct = driver.wlanMeas.multiEval.ofdma.info.fetch()
```

Queries OFDMA common information.

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.3.2 Uinfo<User>

RepCap Settings

```
# Range: Nr1 .. Nr144
rc = driver.wlanMeas.multiEval.ofdma.uinfo.repcap_user_get()
driver.wlanMeas.multiEval.ofdma.uinfo.repcap_user_set(repcap.User.Nr1)
```

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:OFDMa:UINFo<user>
```

class UinfoCls

Uinfo commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: User, default value after init: User.Nr1

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Ru_Count: int: Index of RUs of all sizes (users with STA-ID 2046 are included) .
- Ru_Index: int: Index of the RUs only with the used size RUSize
- Ru_26_Index: int: Index based on RU26
- Ru_Size: int: RU size allocated by the user
- Mcs: int: Modulation and coding scheme
- Dcm: int: The value of DCM field
- Sta_Id: int: The value of the STA-ID field
- No_Of_Sts: int: The value of NSTS field
- Tx_Bf: int: The value of TxBF field
- Coding: enums.CodingType: Coding type

fetch(*user=User.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:OFDMa:UINFo<user>
value: FetchStruct = driver.wlanMeas.multiEval.ofdma.uinfo.fetch(user = repcap.
↪User.Default)
```

Queries OFDMA user-specific information signaled in a HE signal field.

param user
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Uinfo')

return
structure: for return value, see the help for FetchStruct structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.ofdma.uinfo.clone()
```

6.4.1.4 Power

class PowerCls

Power commands group definition. 12 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.power.clone()
```

Subgroups

6.4.1.4.1 Runit<ResourceUnit>

RepCap Settings

```
# Range: Nr1 .. Nr144
rc = driver.wlanMeas.multiEval.power.runit.repcap_resourceUnit_get()
driver.wlanMeas.multiEval.power.runit.repcap_resourceUnit_set(repcap.ResourceUnit.Nr1)
```

class RunitCls

Runit commands group definition. 8 total commands, 5 Subgroups, 0 group commands Repeated Capability: ResourceUnit, default value after init: ResourceUnit.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.power.runit.clone()
```

Subgroups

6.4.1.4.1.1 Average

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:POWer:RUNit<ru>:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*resourceUnit=ResourceUnit.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:POWer:RUNit<ru>:AVERage
value: List[float] = driver.wlanMeas.multiEval.power.runit.average.
↪fetch(resourceUnit = repcap.ResourceUnit.Default)
```

Returns single power value measured for RU at all antennas (OFDMA) .

Suppressed linked return values: reliability

param resourceUnit
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Runit')

return
power_vs_ru_all_antennas: Power vs RU for all antennas

6.4.1.4.1.2 Current

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:POWer:RUNit<ru>:CURRent
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*resourceUnit=ResourceUnit.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:POWer:RUNit<ru>:CURRent
value: List[float] = driver.wlanMeas.multiEval.power.runit.current.
↪fetch(resourceUnit = repcap.ResourceUnit.Default)
```

Returns single power value measured for RU at all antennas (OFDMA) .

Suppressed linked return values: reliability

param resourceUnit
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Runit')

return
power_vs_ru_all_antennas: Power vs RU for all antennas

6.4.1.4.1.3 Maximum

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:POWer:RUNit<ru>:MAXimum
```

class MaximumCls

Maximum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*resourceUnit=ResourceUnit.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:POWer:RUNit<ru>:MAXimum
value: List[float] = driver.wlanMeas.multiEval.power.runit.maximum.
↪fetch(resourceUnit = repcap.ResourceUnit.Default)
```

Returns single power value measured for RU at all antennas (OFDMA) .

Suppressed linked return values: reliability

param resourceUnit
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Runit')

return
power_vs_ru_all_antennas: Power vs RU for all antennas

6.4.1.4.1.4 RxAntenna<RxAntenna>

RepCap Settings

```
# Range: Nr1 .. Nr8
rc = driver.wlanMeas.multiEval.power.runit.rxAntenna.repcap_rxAntenna_get()
driver.wlanMeas.multiEval.power.runit.rxAntenna.repcap_rxAntenna_set(repcap.RxAntenna.
↪Nr1)
```

class RxAntennaCls

RxAntenna commands group definition. 4 total commands, 4 Subgroups, 0 group commands Repeated Capability: RxAntenna, default value after init: RxAntenna.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.power.runit.rxAntenna.clone()
```

Subgroups

6.4.1.4.1.5 Average

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:POWer:RUNit<ru>:RXANtenna<n>:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*resourceUnit=ResourceUnit.Default*, *rxAntenna=RxAntenna.Default*) → float

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:POWer:RUNit<ru>:RXANtenna
↪<n>:AVERage
value: float = driver.wlanMeas.multiEval.power.runit.rxAntenna.average.
↪fetch(resourceUnit = repcap.ResourceUnit.Default, rxAntenna = repcap.
↪RxAntenna.Default)
```

Returns single power value measured for RU at the specified antenna (OFDMA) .

Suppressed linked return values: reliability

param resourceUnit
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Runit')

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

return
power_vs_antenna_vs_ru: Power vs RU vs antenna

6.4.1.4.1.6 Current

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:POWer:RUNit<ru>:RXANtenna<n>:CURRent
```

`class CurrentCls`

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

`fetch`(*resourceUnit=ResourceUnit.Default*, *rxAntenna=RxAntenna.Default*) → float

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:POWer:RUNit<ru>:RXANtenna
↪<n>:CURRent
value: float = driver.wlanMeas.multiEval.power.runit.rxAntenna.current.
↪fetch(resourceUnit = repcap.ResourceUnit.Default, rxAntenna = repcap.
↪RxAntenna.Default)
```

Returns single power value measured for RU at the specified antenna (OFDMA) .

Suppressed linked return values: reliability

param resourceUnit
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Runit')

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

return
power_vs_antenna_vs_ru: Power vs RU vs antenna

6.4.1.4.1.7 Maximum

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:POWer:RUNit<ru>:RXANtenna<n>:MAXimum
```

class MaximumCls

Maximum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*resourceUnit=ResourceUnit.Default*, *rxAntenna=RxAntenna.Default*) → float

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:POWer:RUNit<ru>:RXANtenna
↪<n>:MAXimum
value: float = driver.wlanMeas.multiEval.power.runit.rxAntenna.maximum.
↪fetch(resourceUnit = repcap.ResourceUnit.Default, rxAntenna = repcap.
↪RxAntenna.Default)
```

Returns single power value measured for RU at the specified antenna (OFDMA) .

Suppressed linked return values: reliability

param resourceUnit
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Runit')

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

return
power_vs_antenna_vs_ru: Power vs RU vs antenna

6.4.1.4.1.8 StandardDev

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:POWer:RUNit<ru>:RXANtenna<n>:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*resourceUnit=ResourceUnit.Default*, *rxAntenna=RxAntenna.Default*) → float

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:POWer:RUNit<ru>:RXANtenna
↪<n>:SDEViation
value: float = driver.wlanMeas.multiEval.power.runit.rxAntenna.standardDev.
↪fetch(resourceUnit = repcap.ResourceUnit.Default, rxAntenna = repcap.
↪RxAntenna.Default)
```

Returns single power value measured for RU at the specified antenna (OFDMA) .

Suppressed linked return values: reliability

param resourceUnit
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Runit')

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

return
power_vs_antenna_vs_ru: Power vs RU vs antenna

6.4.1.4.1.9 StandardDev

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:POWer:RUNit<ru>:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*resourceUnit=ResourceUnit.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:POWer:RUNit<ru>:SDEViation
value: List[float] = driver.wlanMeas.multiEval.power.runit.standardDev.
↪fetch(resourceUnit = repcap.ResourceUnit.Default)
```

Returns single power value measured for RU at all antennas (OFDMA) .

Suppressed linked return values: reliability

param resourceUnit
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Runit')

return
power_vs_ru_all_antennas: Power vs RU for all antennas

6.4.1.4.2 RxAntenna<RxAntenna>

RepCap Settings

```
# Range: Nr1 .. Nr8
rc = driver.wlanMeas.multiEval.power.rxAntenna.repcap_rxAntenna_get()
driver.wlanMeas.multiEval.power.rxAntenna.repcap_rxAntenna_set(repcap.RxAntenna.Nr1)
```

class RxAntennaCls

RxAntenna commands group definition. 4 total commands, 4 Subgroups, 0 group commands Repeated Capability: RxAntenna, default value after init: RxAntenna.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.power.rxAntenna.clone()
```

Subgroups

6.4.1.4.2.1 Average

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:POWer:RXANtenna<n>:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*rxAntenna=RxAntenna.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:POWer:RXANtenna<n>:AVERage
value: List[float] = driver.wlanMeas.multiEval.power.rxAntenna.average.
↪fetch(rxAntenna = repcap.RxAntenna.Default)
```

Returns single power value measured at the specified antenna for all RUs (OFDMA) .

Suppressed linked return values: reliability

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

return
power_vs_antenna_all_rus: Power vs antenna for all RUs

6.4.1.4.2.2 Current

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:POWer:RXANtenna<n>:CURRent
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*rxAntenna=RxAntenna.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:POWer:RXANtenna<n>:CURRent
value: List[float] = driver.wlanMeas.multiEval.power.rxAntenna.current.
↪fetch(rxAntenna = repcap.RxAntenna.Default)
```

Returns single power value measured at the specified antenna for all RUs (OFDMA) .

Suppressed linked return values: reliability

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

return
power_vs_antenna_all_rus: Power vs antenna for all RUs

6.4.1.4.2.3 Maximum

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:POWer:RXANtenna<n>:MAXimum
```

class MaximumCls

Maximum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*rxAntenna=RxAntenna.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:POWer:RXANtenna<n>:MAXimum
value: List[float] = driver.wlanMeas.multiEval.power.rxAntenna.maximum.
↪fetch(rxAntenna = repcap.RxAntenna.Default)
```

Returns single power value measured at the specified antenna for all RUs (OFDMA) .

Suppressed linked return values: reliability

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

return
power_vs_antenna_all_rus: Power vs antenna for all RUs

6.4.1.4.2.4 StandardDev

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:POWer:RXANtenna<n>:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*rxAntenna=RxAntenna.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:POWer:RXANtenna<n>
↪:SDEViation
value: List[float] = driver.wlanMeas.multiEval.power.rxAntenna.standardDev.
↪fetch(rxAntenna = repcap.RxAntenna.Default)
```

Returns single power value measured at the specified antenna for all RUs (OFDMA) .

Suppressed linked return values: reliability

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Rx-Antenna')

return
power_vs_antenna_all_rus: Power vs antenna for all RUs

6.4.1.5 PowerVsTime

class PowerVsTimeCls

PowerVsTime commands group definition. 54 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.powerVsTime.clone()
```

Subgroups

6.4.1.5.1 FallingEdge

class FallingEdgeCls

FallingEdge commands group definition. 9 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.powerVsTime.fallingEdge.clone()
```

Subgroups

6.4.1.5.1.1 Average

SCPI Commands :

```
CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:FEDGe:AVERage
READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:FEDGe:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:FEDGe:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Power: enums.ResultStatus2: No parameter help available

- Out_Of_Tol: enums.ResultStatus2: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for PVT measurements exceeding the specified power limit

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Power: float: No parameter help available
- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for PVT measurements exceeding the specified power limit

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:FEDGe:AVERage
value: CalculateStruct = driver.wlanMeas.multiEval.powerVsTime.fallingEdge.
↪average.calculate()
```

Returns the current, average and maximum ramp durations of the power vs time measurement, for the falling edge (FEDGe) and rising edge (REDGe) . The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:FEDGe:AVERage
value: ResultData = driver.wlanMeas.multiEval.powerVsTime.fallingEdge.average.
↪fetch()
```

Returns the current, average and maximum ramp durations of the power vs time measurement, for the falling edge (FEDGe) and rising edge (REDGe) . The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:FEDGe:AVERage
value: ResultData = driver.wlanMeas.multiEval.powerVsTime.fallingEdge.average.
↪read()
```

Returns the current, average and maximum ramp durations of the power vs time measurement, for the falling edge (FEDGe) and rising edge (REDGe) . The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.5.1.2 Current

SCPI Commands :

```
CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:FEDGe:CURRent
READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:FEDGe:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:FEDGe:CURRent
```

class CurrentCls

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Power: enums.ResultStatus2: No parameter help available
- Out_Of_Tol: enums.ResultStatus2: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for PVT measurements exceeding the specified power limit

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Power: float: No parameter help available
- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for PVT measurements exceeding the specified power limit

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:FEDGe:CURRent
value: CalculateStruct = driver.wlanMeas.multiEval.powerVsTime.fallingEdge.
↪current.calculate()
```

Returns the current, average and maximum ramp durations of the power vs time measurement, for the falling edge (FEDGe) and rising edge (REDGe) . The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:FEDGe:CURRent
value: ResultData = driver.wlanMeas.multiEval.powerVsTime.fallingEdge.current.
↪fetch()
```

Returns the current, average and maximum ramp durations of the power vs time measurement, for the falling edge (FEDGe) and rising edge (REDGe) . The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:FEDGe:CURRent
value: ResultData = driver.wlanMeas.multiEval.powerVsTime.fallingEdge.current.
↪read()
```

Returns the current, average and maximum ramp durations of the power vs time measurement, for the falling edge (FEDGe) and rising edge (REDGe) . The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.5.1.3 Maximum

SCPI Commands :

```
CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:FEDGe:MAXimum
READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:FEDGe:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:FEDGe:MAXimum
```

class MaximumCls

Maximum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Power: enums.ResultStatus2: No parameter help available
- Out_Of_Tol: enums.ResultStatus2: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for PVT measurements exceeding the specified power limit

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Power: float: No parameter help available
- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for PVT measurements exceeding the specified power limit

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:FEDGe:MAXimum
value: CalculateStruct = driver.wlanMeas.multiEval.powerVsTime.fallingEdge.
↪maximum.calculate()
```

Returns the current, average and maximum ramp durations of the power vs time measurement, for the falling edge (FEDGe) and rising edge (REDGe) . The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:FEDGe:MAXimum
value: ResultData = driver.wlanMeas.multiEval.powerVsTime.fallingEdge.maximum.
↪fetch()
```

Returns the current, average and maximum ramp durations of the power vs time measurement, for the falling edge (FEDGe) and rising edge (REDGe) . The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:FEDGe:MAXimum
value: ResultData = driver.wlanMeas.multiEval.powerVsTime.fallingEdge.maximum.
↪read()
```

Returns the current, average and maximum ramp durations of the power vs time measurement, for the falling edge (FEDGe) and rising edge (REDGe) . The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.5.2 Mimo<Mimo>

RepCap Settings

```
# Range: Nr1 .. Nr8
rc = driver.wlanMeas.multiEval.powerVsTime.mimo.repcap_mimo_get()
driver.wlanMeas.multiEval.powerVsTime.mimo.repcap_mimo_set(repcap.Mimo.Nr1)
```

class MimoCls

Mimo commands group definition. 3 total commands, 1 Subgroups, 0 group commands Repeated Capability: Mimo, default value after init: Mimo.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.powerVsTime.mimo.clone()
```

Subgroups

6.4.1.5.2.1 TeDistribution

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:MIMO<n>:TEDistrib
FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:MIMO<n>:TEDistrib
CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:MIMO<n>:TEDistrib
```

class TeDistributionCls

TeDistribution commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Te_Percentage: enums.ResultStatus2: Percentage of TEs
- Te_Outside: enums.ResultStatus2: Number of detected TEs
- Te_Total: enums.ResultStatus2: Number of measured values

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Te_Percentage: float: Percentage of TEs
- Te_Outside: int: Number of detected TEs
- Te_Total: int: Number of measured values

calculate(*mimo=Mimo.Default*) → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:MIMO<n>
↪:TEDistrib
value: CalculateStruct = driver.wlanMeas.multiEval.powerVsTime.mimo.
↪teDistribution.calculate(mimo = repcap.Mimo.Default)
```

Return the scalar results for timing error (TE) distribution for MIMO. The commands are only supported for OFDM standards. Exceeding the limit has no impact on the stop On Limit Failure condition or out of tolerance counter.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch(*mimo=Mimo.Default*) → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:MIMO<n>:TEDistrib
value: ResultData = driver.wlanMeas.multiEval.powerVsTime.mimo.teDistribution.
↪fetch(mimo = repcap.Mimo.Default)
```

Return the scalar results for timing error (TE) distribution for MIMO. The commands are only supported for OFDM standards. Exceeding the limit has no impact on the stop On Limit Failure condition or out of tolerance counter.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

read(*mimo=Mimo.Default*) → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:MIMO<n>:TEDistrib
value: ResultData = driver.wlanMeas.multiEval.powerVsTime.mimo.teDistribution.
↪read(mimo = repcap.Mimo.Default)
```

Return the scalar results for timing error (TE) distribution for MIMO. The commands are only supported for OFDM standards. Exceeding the limit has no impact on the stop On Limit Failure condition or out of tolerance counter.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.5.3 RisingEdge

class RisingEdgeCls

RisingEdge commands group definition. 9 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.powerVsTime.risingEdge.clone()
```

Subgroups

6.4.1.5.3.1 Average

SCPI Commands :

```
CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:REDGe:AVERage
READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:REDGe:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:REDGe:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Power: enums.ResultStatus2: No parameter help available
- Out_Of_Tol: enums.ResultStatus2: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for PVT measurements exceeding the specified power limit

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Power: float: No parameter help available
- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for PVT measurements exceeding the specified power limit

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:REDGe:AVERage
value: CalculateStruct = driver.wlanMeas.multiEval.powerVsTime.risingEdge.
↪average.calculate()
```

Returns the current, average and maximum ramp durations of the power vs time measurement, for the falling edge (FEDGe) and rising edge (REDGe) . The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:REDGe:AVERage
value: ResultData = driver.wlanMeas.multiEval.powerVsTime.risingEdge.average.
↪fetch()
```

Returns the current, average and maximum ramp durations of the power vs time measurement, for the falling edge (FEDGe) and rising edge (REDGe) . The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:REDGe:AVERage
value: ResultData = driver.wlanMeas.multiEval.powerVsTime.risingEdge.average.
↪read()
```

Returns the current, average and maximum ramp durations of the power vs time measurement, for the falling edge (FEDGe) and rising edge (REDGe) . The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.5.3.2 Current

SCPI Commands :

```
CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:REDGe:CURRent
READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:REDGe:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:REDGe:CURRent
```

class CurrentCls

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Power: enums.ResultStatus2: No parameter help available
- Out_Of_Tol: enums.ResultStatus2: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for PVT measurements exceeding the specified power limit

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Power: float: No parameter help available
- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for PVT measurements exceeding the specified power limit

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:REDGe:CURRent
value: CalculateStruct = driver.wlanMeas.multiEval.powerVsTime.risingEdge.
↪current.calculate()
```

Returns the current, average and maximum ramp durations of the power vs time measurement, for the falling edge (FEDGe) and rising edge (REDGe) . The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:REDGe:CURRent
value: ResultData = driver.wlanMeas.multiEval.powerVsTime.risingEdge.current.
↪fetch()
```

Returns the current, average and maximum ramp durations of the power vs time measurement, for the falling edge (FEDGe) and rising edge (REDGe) . The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:REDGe:CURRent
value: ResultData = driver.wlanMeas.multiEval.powerVsTime.risingEdge.current.
↪read()
```

Returns the current, average and maximum ramp durations of the power vs time measurement, for the falling edge (FEDGe) and rising edge (REDGe) . The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.5.3.3 Maximum

SCPI Commands :

```
CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:REDGe:MAXimum
READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:REDGe:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:REDGe:MAXimum
```

class MaximumCls

Maximum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Power: enums.ResultStatus2: No parameter help available
- Out_Of_Tol: enums.ResultStatus2: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for PVT measurements exceeding the specified power limit

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Power: float: No parameter help available
- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for PVT measurements exceeding the specified power limit

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:REDGe:MAXimum
value: CalculateStruct = driver.wlanMeas.multiEval.powerVsTime.risingEdge.
↪maximum.calculate()
```

Returns the current, average and maximum ramp durations of the power vs time measurement, for the falling edge (FEDGe) and rising edge (REDGe) . The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:REDGe:MAXimum
value: ResultData = driver.wlanMeas.multiEval.powerVsTime.risingEdge.maximum.
↪fetch()
```

Returns the current, average and maximum ramp durations of the power vs time measurement, for the falling edge (FEDGe) and rising edge (REDGe) . The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:REDGe:MAXimum
value: ResultData = driver.wlanMeas.multiEval.powerVsTime.risingEdge.maximum.
↪read()
```

Returns the current, average and maximum ramp durations of the power vs time measurement, for the falling edge (FEDGe) and rising edge (REDGe) . The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.5.4 TeDistribution

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:TEDistrib
FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:TEDistrib
CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:TEDistrib
```

class TeDistributionCls

TeDistribution commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Te_Percentage: enums.ResultStatus2: Percentage of TEs
- Te_Outside: enums.ResultStatus2: Number of detected TEs
- Te_Total: enums.ResultStatus2: Number of measured values

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Te_Percentage: float: Percentage of TEs
- Te_Outside: int: Number of detected TEs
- Te_Total: int: Number of measured values

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:TEDistrib
value: CalculateStruct = driver.wlanMeas.multiEval.powerVsTime.teDistribution.
↪calculate()
```

Return the scalar results for timing error (TE) distribution. The commands are only supported for OFDM standards. Exceeding the limit has no impact on the stop On Limit Failure condition or out of tolerance counter.

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:TEDistrib
value: ResultData = driver.wlanMeas.multiEval.powerVsTime.teDistribution.fetch()
```

Return the scalar results for timing error (TE) distribution. The commands are only supported for OFDM standards. Exceeding the limit has no impact on the stop On Limit Failure condition or out of tolerance counter.

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:TEDistrib
value: ResultData = driver.wlanMeas.multiEval.powerVsTime.teDistribution.read()
```

Return the scalar results for timing error (TE) distribution. The commands are only supported for OFDM standards. Exceeding the limit has no impact on the stop On Limit Failure condition or out of tolerance counter.

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.5.5 Terror

class TerrorCls

Terror commands group definition. 30 total commands, 6 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.powerVsTime.terror.clone()
```

Subgroups

6.4.1.5.5.1 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:AVERage
CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate() → ResultStatus2

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:AVERage
value: enums.ResultStatus2 = driver.wlanMeas.multiEval.powerVsTime.terror.
↪average.calculate()
```

Return the current, average, minimum, maximum and standard deviation timing error single value results of the power vs time measurement. The commands are only supported for OFDM standards. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

return
timing_error_aver: No help available

fetch() → float

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:AVERage
value: float = driver.wlanMeas.multiEval.powerVsTime.terror.average.fetch()
```

Return the current, average, minimum, maximum and standard deviation timing error single value results of the power vs time measurement. The commands are only supported for OFDM standards. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

return
timing_error_aver: No help available

read() → float

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:AVERage
value: float = driver.wlanMeas.multiEval.powerVsTime.terror.average.read()
```

Return the current, average, minimum, maximum and standard deviation timing error single value results of the power vs time measurement. The commands are only supported for OFDM standards. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

return
timing_error_aver: No help available

6.4.1.5.5.2 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:CURRent
CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:CURRent
```

class CurrentCls

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate() → ResultStatus2

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:CURRent
value: enums.ResultStatus2 = driver.wlanMeas.multiEval.powerVsTime.terror.
↪current.calculate()
```

Return the current, average, minimum, maximum and standard deviation timing error single value results of the power vs time measurement. The commands are only supported for OFDM standards. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

return
timing_error_curr: No help available

fetch() → float

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:CURRent
value: float = driver.wlanMeas.multiEval.powerVsTime.terror.current.fetch()
```

Return the current, average, minimum, maximum and standard deviation timing error single value results of the power vs time measurement. The commands are only supported for OFDM standards. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

return
timing_error_curr: No help available

read() → float

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:CURRent
value: float = driver.wlanMeas.multiEval.powerVsTime.terror.current.read()
```

Return the current, average, minimum, maximum and standard deviation timing error single value results of the power vs time measurement. The commands are only supported for OFDM standards. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

return
timing_error_curr: No help available

6.4.1.5.5.3 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:MAXimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:MAXimum
```

class MaximumCls

Maximum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate() → ResultStatus2

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:MAXimum
value: enums.ResultStatus2 = driver.wlanMeas.multiEval.powerVsTime.terror.
↪maximum.calculate()
```

Return the current, average, minimum, maximum and standard deviation timing error single value results of the power vs time measurement. The commands are only supported for OFDM standards. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

return
timing_error_max: No help available

fetch() → float

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:MAXimum
value: float = driver.wlanMeas.multiEval.powerVsTime.terror.maximum.fetch()
```

Return the current, average, minimum, maximum and standard deviation timing error single value results of the power vs time measurement. The commands are only supported for OFDM standards. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

return
timing_error_max: No help available

read() → float

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:MAXimum
value: float = driver.wlanMeas.multiEval.powerVsTime.terror.maximum.read()
```

Return the current, average, minimum, maximum and standard deviation timing error single value results of the power vs time measurement. The commands are only supported for OFDM standards. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

return
timing_error_max: No help available

6.4.1.5.5.4 Mimo<Mimo>

RepCap Settings

```
# Range: Nr1 .. Nr8
rc = driver.wlanMeas.multiEval.powerVsTime.terror.mimo.repcap_mimo_get()
driver.wlanMeas.multiEval.powerVsTime.terror.mimo.repcap_mimo_set(repcap.Mimo.Nr1)
```

class MimoCls

Mimo commands group definition. 15 total commands, 5 Subgroups, 0 group commands Repeated Capability: Mimo, default value after init: Mimo.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.powerVsTime.terror.mimo.clone()
```

Subgroups

6.4.1.5.5.5 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:MIMO<n>:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:MIMO<n>:AVERage
CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:MIMO<n>:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*mimo=Mimo.Default*) → ResultStatus2

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:MIMO<n>
↪:AVERage
value: enums.ResultStatus2 = driver.wlanMeas.multiEval.powerVsTime.terror.mimo.
↪average.calculate(mimo = repcap.Mimo.Default)
```

Return the current, average, minimum, maximum and standard deviation timing error single value results of the power vs time MIMO measurement. The commands are only supported for OFDM standards. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
timing_error_avg: No help available

fetch(*mimo=Mimo.Default*) → float

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:MIMO<n>
↪:AVERage
value: float = driver.wlanMeas.multiEval.powerVsTime.terror.mimo.average.
↪fetch(mimo = repcap.Mimo.Default)
```

Return the current, average, minimum, maximum and standard deviation timing error single value results of the power vs time MIMO measurement. The commands are only supported for OFDM standards. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
timing_error_avg: No help available

read(*mimo=Mimo.Default*) → float

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:MIMO<n>
↪:AVERage
value: float = driver.wlanMeas.multiEval.powerVsTime.terror.mimo.average.
↪read(mimo = repcap.Mimo.Default)
```

Return the current, average, minimum, maximum and standard deviation timing error single value results of the power vs time MIMO measurement. The commands are only supported for OFDM standards. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
timing_error_avg: No help available

6.4.1.5.5.6 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:MIMO<n>:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:MIMO<n>:CURRent
CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:MIMO<n>:CURRent
```

class CurrentCls

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*mimo=Mimo.Default*) → ResultStatus2

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:MIMO<n>
↪:CURRent
value: enums.ResultStatus2 = driver.wlanMeas.multiEval.powerVsTime.terror.mimo.
↪current.calculate(mimo = repcap.Mimo.Default)
```

Return the current, average, minimum, maximum and standard deviation timing error single value results of the power vs time MIMO measurement. The commands are only supported for OFDM standards. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
timing_error_curr: No help available

fetch(*mimo=Mimo.Default*) → float

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:MIMO<n>
↪:CURRent
value: float = driver.wlanMeas.multiEval.powerVsTime.terror.mimo.current.
↪fetch(mimo = repcap.Mimo.Default)
```

Return the current, average, minimum, maximum and standard deviation timing error single value results of the power vs time MIMO measurement. The commands are only supported for OFDM standards. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
timing_error_curr: No help available

read(*mimo=Mimo.Default*) → float

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:MIMO<n>
↪:CURRent
value: float = driver.wlanMeas.multiEval.powerVsTime.terror.mimo.current.
↪read(mimo = repcap.Mimo.Default)
```

Return the current, average, minimum, maximum and standard deviation timing error single value results of the power vs time MIMO measurement. The commands are only supported for OFDM standards. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
timing_error_curr: No help available

6.4.1.5.5.7 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:MIMO<n>:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:MIMO<n>:MAXimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:MIMO<n>:MAXimum
```

class MaximumCls

Maximum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*mimo=Mimo.Default*) → ResultStatus2

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:MIMO<n>
↪:MAXimum
value: enums.ResultStatus2 = driver.wlanMeas.multiEval.powerVsTime.terror.mimo.
↪maximum.calculate(mimo = repcap.Mimo.Default)
```

Return the current, average, minimum, maximum and standard deviation timing error single value results of the power vs time MIMO measurement. The commands are only supported for OFDM standards. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
timing_error_max: No help available

fetch(*mimo=Mimo.Default*) → float

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:MIMO<n>
↪:MAXimum
value: float = driver.wlanMeas.multiEval.powerVsTime.terror.mimo.maximum.
↪fetch(mimo = repcap.Mimo.Default)
```

Return the current, average, minimum, maximum and standard deviation timing error single value results of the power vs time MIMO measurement. The commands are only supported for OFDM standards. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
timing_error_max: No help available

read(*mimo=Mimo.Default*) → float

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:MIMO<n>
↪:MAXimum
value: float = driver.wlanMeas.multiEval.powerVsTime.terror.mimo.maximum.
↪read(mimo = repcap.Mimo.Default)
```

Return the current, average, minimum, maximum and standard deviation timing error single value results of the power vs time MIMO measurement. The commands are only supported for OFDM standards. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
timing_error_max: No help available

6.4.1.5.5.8 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:MIMO<n>:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:MIMO<n>:MINimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:MIMO<n>:MINimum
```

class MinimumCls

Minimum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*mimo=Mimo.Default*) → ResultStatus2

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:MIMO<n>
↪:MINimum
value: enums.ResultStatus2 = driver.wlanMeas.multiEval.powerVsTime.terror.mimo.
↪minimum.calculate(mimo = repcap.Mimo.Default)
```

Return the current, average, minimum, maximum and standard deviation timing error single value results of the power vs time MIMO measurement. The commands are only supported for OFDM standards. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
timing_error_min: No help available

fetch(*mimo=Mimo.Default*) → float

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:MIMO<n>
↪:MINimum
value: float = driver.wlanMeas.multiEval.powerVsTime.terror.mimo.minimum.
↪fetch(mimo = repcap.Mimo.Default)
```

No command help available

Suppressed linked return values: reliability

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
timing_error_min: No help available

read(*mimo=Mimo.Default*) → float

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:MIMO<n>
↪:MINimum
value: float = driver.wlanMeas.multiEval.powerVsTime.terror.mimo.minimum.
↪read(mimo = repcap.Mimo.Default)
```

Return the current, average, minimum, maximum and standard deviation timing error single value results of the power vs time MIMO measurement. The commands are only supported for OFDM standards. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
timing_error_min: No help available

6.4.1.5.5.9 StandardDev

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:MIMO<n>:SDEViation
FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:MIMO<n>:SDEViation
CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:MIMO<n>:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*mimo=Mimo.Default*) → ResultStatus2

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:MIMO<n>
↪:SDEViation
value: enums.ResultStatus2 = driver.wlanMeas.multiEval.powerVsTime.terror.mimo.
↪standardDev.calculate(mimo = repcap.Mimo.Default)
```

Return the current, average, minimum, maximum and standard deviation timing error single value results of the power vs time MIMO measurement. The commands are only supported for OFDM standards. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
timing_error_sdev: No help available

fetch(*mimo=Mimo.Default*) → float

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:MIMO<n>
↪:SDEViation
value: float = driver.wlanMeas.multiEval.powerVsTime.terror.mimo.standardDev.
↪fetch(mimo = repcap.Mimo.Default)
```

Return the current, average, minimum, maximum and standard deviation timing error single value results of the power vs time MIMO measurement. The commands are only supported for OFDM standards. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
timing_error_sdev: No help available

read(*mimo=Mimo.Default*) → float

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:MIMO<n>
↪:SDEViation
value: float = driver.wlanMeas.multiEval.powerVsTime.terror.mimo.standardDev.
↪read(mimo = repcap.Mimo.Default)
```

Return the current, average, minimum, maximum and standard deviation timing error single value results of the power vs time MIMO measurement. The commands are only supported for OFDM standards. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
timing_error_sdev: No help available

6.4.1.5.5.10 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:MINimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:MINimum
```

class MinimumCls

Minimum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate() → ResultStatus2

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:MINimum
value: enums.ResultStatus2 = driver.wlanMeas.multiEval.powerVsTime.terror.
↪minimum.calculate()
```

Return the current, average, minimum, maximum and standard deviation timing error single value results of the power vs time measurement. The commands are only supported for OFDM standards. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

return
timing_error_min: No help available

fetch() → float

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:MINimum
value: float = driver.wlanMeas.multiEval.powerVsTime.terror.minimum.fetch()
```

Return the current, average, minimum, maximum and standard deviation timing error single value results of the power vs time measurement. The commands are only supported for OFDM standards. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

return
timing_error_min: No help available

read() → float

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:MINimum
value: float = driver.wlanMeas.multiEval.powerVsTime.terror.minimum.read()
```

Return the current, average, minimum, maximum and standard deviation timing error single value results of the power vs time measurement. The commands are only supported for OFDM standards. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

return
timing_error_min: No help available

6.4.1.5.5.11 StandardDev

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:SDEViation
FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:SDEViation
CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate() → ResultStatus2

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>
↪:MEValuation:PVTime:TERRor:SDEViation
value: enums.ResultStatus2 = driver.wlanMeas.multiEval.powerVsTime.terror.
↪standardDev.calculate()
```

Return the current, average, minimum, maximum and standard deviation timing error single value results of the power vs time measurement. The commands are only supported for OFDM standards. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

return
timing_error_sdev: No help available

fetch() → float

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:SDEViation
value: float = driver.wlanMeas.multiEval.powerVsTime.terror.standardDev.fetch()
```

Return the current, average, minimum, maximum and standard deviation timing error single value results of the power vs time measurement. The commands are only supported for OFDM standards. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

return
timing_error_sdev: No help available

read() → float

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:TERRor:SDEViation
value: float = driver.wlanMeas.multiEval.powerVsTime.terror.standardDev.read()
```

Return the current, average, minimum, maximum and standard deviation timing error single value results of the power vs time measurement. The commands are only supported for OFDM standards. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

return
timing_error_sdev: No help available

6.4.1.6 Sinfo

class SinfoCls

Sinfo commands group definition. 95 total commands, 7 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.sinfo.clone()
```

Subgroups

6.4.1.6.1 Heb

class HebCls

Heb commands group definition. 14 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.sinfo.heb.clone()
```

Subgroups

6.4.1.6.1.1 Channel<Channel>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.wlanMeas.multiEval.sinfo.heb.channel.repcap_channel_get()
driver.wlanMeas.multiEval.sinfo.heb.channel.repcap_channel_set(repcap.Channel.Nr1)
```

class ChannelCls

Channel commands group definition. 14 total commands, 2 Subgroups, 0 group commands Repeated Capability: Channel, default value after init: Channel.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.sinfo.heb.channel.clone()
```

Subgroups

6.4.1.6.1.2 Cfield

class CfieldCls

Cfield commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.sinfo.heb.channel.cfield.clone()
```

Subgroups

6.4.1.6.1.3 Crc

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEB:CHANnel<ch_index>:CFIeld:CRC
```

class CrcCls

Crc commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch(*channel=Channel.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEB:CHANnel<ch_index>
↪:CFIeld:CRC
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.heb.channel.cfield.crc.
↪fetch(channel = repcap.Channel.Default)
```

Queries the value of the CRC field signaled for the channel in HE-SIG-B common field for multi-user MIMO.

param channel
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Channel')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.1.4 Cru

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEB:CHANnel<ch_index>:CFIeld:CRU
```

class CruCls

Cru commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch(*channel=Channel.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEB:CHANnel<ch_index>
↪:CFIeld:CRU
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.heb.channel.cfield.cru.
↪fetch(channel = repcap.Channel.Default)
```

Queries the value of the Center 26-tone RU field signaled for the channel in HE-SIG-B common field for multi-user MIMO.

param channel
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Channel')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.1.5 RuAllocation

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEB:CHANnel<ch_index>
↪:CFIeld:RUALlocation
```

class RuAllocationCls

RuAllocation commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch(*channel=Channel.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEB:CHANnel<ch_index>
↪:CFIeld:RUALlocation
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.heb.channel.cfield.
↪ruAllocation.fetch(channel = repcap.Channel.Default)
```

Queries the value of the RU Allocation field signaled for the channel in HE-SIG-B common field for multi-user MIMO.

param channel
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Channel')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.1.6 Tail

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEB:CHANnel<ch_index>:CFIeld:TAIL
```

class TailCls

Tail commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch(*channel=Channel.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEB:CHANnel<ch_index>
↪:CFIeld:TAIL
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.heb.channel.cfield.tail.
↪fetch(channel = repcap.Channel.Default)
```

Queries the value of the Tail field signaled for the channel in HE-SIG-B common field for multi-user MIMO.

param channel
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Channel')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.1.7 Ufield<UserIx>

RepCap Settings

```
# Range: Nr1 .. Nr144
rc = driver.wlanMeas.multiEval.sinfo.heb.channel.ufield.repcap_userIx_get()
driver.wlanMeas.multiEval.sinfo.heb.channel.ufield.repcap_userIx_set(repcap.UserIx.Nr1)
```

class UfieldCls

Ufield commands group definition. 10 total commands, 10 Subgroups, 0 group commands Repeated Capability: UserIx, default value after init: UserIx.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.sinfo.heb.channel.ufield.clone()
```

Subgroups

6.4.1.6.1.8 Coding

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEB:CHANnel<ch_index>:UFIeld<usr_
↪index>:CODing
```

class CodingCls

Coding commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch(*channel=Channel.Default*, *userIx=UserIx.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEB:CHANnel<ch_index>
↪:UFIeld<usr_index>:CODing
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.heb.channel.ufield.coding.
↪fetch(channel = repcap.Channel.Default, userIx = repcap.UserIx.Default)
```

Queries the value of Coding field signaled for the channel and user in HE-SIG-B user-specific field for multi-user MIMO.

param channel
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Channel')

param userIx
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Ufield')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.1.9 Crc

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEB:CHANnel<ch_index>:UFIeld<usr_
↪index>:CRC
```

class CrcCls

Crc commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch(*channel=Channel.Default*, *userIx=UserIx.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEB:CHANnel<ch_index>
↪:UFIeld<usr_index>:CRC
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.heb.channel.ufield.crc.
↪fetch(channel = repcap.Channel.Default, userIx = repcap.UserIx.Default)
```

Queries the value of CRC field signaled for the channel and user in HE-SIG-B user-specific field for multi-user MIMO.

param channel
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Channel')

param userIx
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Ufield')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.1.10 Dcm

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEB:CHANnel<ch_index>:UFIeld<usr_
↪index>:DCM
```

class DcmCls

Dcm commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch(*channel=Channel.Default*, *userIx=UserIx.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEB:CHANnel<ch_index>
↪:UFIeld<usr_index>:DCM
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.heb.channel.ufield.dcm.
↪fetch(channel = repcap.Channel.Default, userIx = repcap.UserIx.Default)
```

Queries the value of DCM field signaled for the channel and user in HE-SIG-B user-specific field for SU-MIMO.

param channel
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Channel')

param userIx
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Ufield')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.1.11 Mcs

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEB:CHANnel<ch_index>:UFIeld<usr_
↪index>:MCS
```

class McsCls

Mcs commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available

- Value_Dec: int: No parameter help available

fetch(*channel=Channel.Default*, *userIx=UserIx.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEB:CHANnel<ch_index>
↪:UFIeld<usr_index>:MCS
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.heb.channel.ufield.mcs.
↪fetch(channel = repcap.Channel.Default, userIx = repcap.UserIx.Default)
```

Queries the value of MCS field signaled for the channel and user in HE-SIG-B user-specific field for MIMO.

param channel
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Channel')

param userIx
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Ufield')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.1.12 Nsts

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEB:CHANnel<ch_index>:UFIeld<usr_
↪index>:NSTS
```

class NstsCls

Nsts commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch(*channel=Channel.Default*, *userIx=UserIx.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEB:CHANnel<ch_index>
↪:UFIeld<usr_index>:NSTS
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.heb.channel.ufield.nsts.
↪fetch(channel = repcap.Channel.Default, userIx = repcap.UserIx.Default)
```

Queries the value of NSTS field signaled for the channel and user in HE-SIG-B user-specific field for SU-MIMO.

param channel
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Channel')

param userIx
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Ufield')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.1.13 Reserved

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEB:CHANnel<ch_index>:UFIeld<usr_
↪index>:REServed
```

class ReservedCls

Reserved commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch(*channel=Channel.Default*, *userIx=UserIx.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEB:CHANnel<ch_index>
↪:UFIeld<usr_index>:REServed
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.heb.channel.ufield.
↪reserved.fetch(channel = repcap.Channel.Default, userIx = repcap.UserIx.
↪Default)
```

Queries the value of Reserved field signaled for the channel and user in HE-SIG-B user-specific field for multi-user MIMO.

param channel
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Channel')

param userIx
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Ufield')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.1.14 SpaConfig

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEB:CHANnel<ch_index>:UFIeld<usr_
↪index>:SPAConfig
```

class SpaConfigCls

SpaConfig commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch(*channel=Channel.Default*, *userIx=UserIx.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEB:CHANnel<ch_index>
↪:UFIeld<usr_index>:SPAConfig
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.heb.channel.ufield.
↪spaConfig.fetch(channel = repcap.Channel.Default, userIx = repcap.UserIx.
↪Default)
```

Queries the value of Spatial Configuration field signaled for the channel and user in HE-SIG-B user-specific field for multi-user MIMO.

param channel
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Channel')

param userIx
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Ufield')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.1.15 StaId

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEB:CHANnel<ch_index>:UFIeld<usr_
↪index>:STAid
```

class StaIdCls

StaId commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch(*channel=Channel.Default*, *userIx=UserIx.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEB:CHANnel<ch_index>
↪:UFIeld<usr_index>:STAid
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.heb.channel.ufield.staId.
↪fetch(channel = repcap.Channel.Default, userIx = repcap.UserIx.Default)
```

Queries the value of the STA-ID field signaled for the channel and user in HE-SIG-B user-specific field for MIMO.

param channel
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Channel')

param userIx
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Ufield')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.1.16 Tail

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEB:CHANnel<ch_index>:UFIeld<usr_
↪index>:TAIL
```

class TailCls

Tail commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch(*channel=Channel.Default*, *userIx=UserIx.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEB:CHANnel<ch_index>
↪:UFIeld<usr_index>:TAIL
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.heb.channel.ufield.tail.
↪fetch(channel = repcap.Channel.Default, userIx = repcap.UserIx.Default)
```

Queries the value of Tail field signaled for the channel and user in HE-SIG-B user-specific field for multi-user MIMO.

param channel
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Channel')

param userIx
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Ufield')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.1.17 TxBeamforming

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEB:CHANnel<ch_index>:UFIeld<usr_
↪index>:TXBeamform
```

class TxBeamformingCls

TxBeamforming commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch(*channel=Channel.Default*, *userIx=UserIx.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEB:CHANnel<ch_index>
↪:UFIeld<usr_index>:TXBeamform
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.heb.channel.ufield.
↪txBeamforming.fetch(channel = repcap.Channel.Default, userIx = repcap.UserIx.
↪Default)
```

Queries the value of Tx Beamforming field signaled for the channel and user in HE-SIG-B user-specific field for SU-MIMO.

param channel
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Channel')

param userIx
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Ufield')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.2 Hemu

class HemuCls

Hemu commands group definition. 19 total commands, 19 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.sinfo.hemu.clone()
```

Subgroups

6.4.1.6.2.1 Bdcm

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:BDCM
```

class BdcmCls

Bdcm commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:BDCM
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hemu.bdcm.fetch()
```

Queries the value of SIGB DCM field signaled in HE signal field for multi-user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.2.2 Bmcs

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:BMCS
```

class BmcsCls

Bmcs commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:BMCS
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hemu.bmcs.fetch()
```

Queries the value of SIGB MCS field signaled in HE signal field for multi-user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.2.3 BssColor

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:BSSColor
```

class BssColorCls

BssColor commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:BSSColor
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hemu.bssColor.fetch()
```

Queries the value of BSS Color field signaled in HE signal field for multi-user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.2.4 Bw

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:BW
```

class BwCls

Bw commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:BW
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hemu.bw.fetch()
```

Queries the value of Bandwidth field signaled in HE signal field for multi-user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.2.5 Crc

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:CRC
```

class CrcCls

Crc commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available
- Check: bool: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:CRC
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hemu.crc.fetch()
```

Queries the value of CRC field signaled in HE signal field for multi-user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.2.6 Doppler

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:DOPPler
```

class DopplerCls

Doppler commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available

- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:DOPPler
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hemu.doppler.fetch()
```

Queries the value of Doppler field signaled in HE signal field for multi-user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.2.7 GiltfSize

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:GILTfsize
```

class GiltfSizeCls

GiltfSize commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:GILTfsize
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hemu.giltfSize.fetch()
```

Queries the value of GI+LTF Size field signaled in HE signal field for multi-user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.2.8 Ldpc

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:LDPC
```

class LdpcCls

Ldpc commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available

- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:LDPC
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hemu.ldpc.fetch()
```

Queries the value of LPDC Extra Symbol Segment field signaled in HE signal field for multi-user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.2.9 NltfSymbols

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:NLTFsymbols
```

class NltfSymbolsCls

NltfSymbols commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:NLTFsymbols
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hemu.nltfSymbols.fetch()
```

Queries the value of Number of HE-LTF Symbols And Midamble Periodicity field signaled in HE signal field for multi-user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.2.10 NsbSymbols

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:NSBSymbols
```

class NsbSymbolsCls

NsbSymbols commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'

- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:NSBSymbols
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hemu.nsbSymbols.fetch()
```

Queries the value of Number Of HE-SIG-B Symbols Or MU-MIMO Users field signaled in the HE signal field for multi-user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.2.11 PeDisambiguity

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:PEDisambig
```

class PeDisambiguityCls

PeDisambiguity commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:PEDisambig
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hemu.peDisambiguity.fetch()
```

Queries the value of PE Disambiguity field signaled in HE signal field for multi-user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.2.12 PfecPadding

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:PFECpadding
```

class PfecPaddingCls

PfecPadding commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:PFECpadding
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hemu.pfecPadding.fetch()
```

Queries the value of Pre-FEC Padding Factor field signaled in HE signal field for multi-user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.2.13 Reserved

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:REServed
```

class ReservedCls

Reserved commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:REServed
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hemu.reserved.fetch()
```

Queries the value of Reserved field signaled in HE signal field for multi-user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.2.14 SbCompress

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:SBCompress
```

class SbCompressCls

SbCompress commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:SBCompress
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hemu.sbCompress.fetch()
```

Queries the value of SIGB Compression field signaled in HE signal field for multi-user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.2.15 SpatialReuse

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:SPATialreuse
```

class SpatialReuseCls

SpatialReuse commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:SPATialreuse
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hemu.spatialReuse.fetch()
```

Queries the value of Spatial Reuse field signaled in HE signal field for multi-user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.2.16 Stbc

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:STBC
```

class StbcCls

Stbc commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:STBC
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hemu.stbc.fetch()
```

Queries the value of STBC field signaled in HE signal field for multi-user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.2.17 Tail

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:TAIL
```

class TailCls

Tail commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available
- Check: bool: Indicates passed or failed check verdict

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:TAIL
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hemu.tail.fetch()
```

Queries the value of Tail field signaled in HE signal field for multi-user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.2.18 TxOp

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:TXOP
```

class TxOpCls

TxOp commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:TXOP
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hemu.txOp.fetch()
```

Queries the value of TXOP field signaled in HE signal field for multi-user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.2.19 UlDl

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:ULDL
```

class UlDlCls

UlDl commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:ULDL
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hemu.ulDl.fetch()
```

Queries the value of UL/DL field signaled in HE signal field for multi-user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.3 Hesu

class HesuCls

Hesu commands group definition. 21 total commands, 21 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.sinfo.hesu.clone()
```

Subgroups

6.4.1.6.3.1 BeamChange

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:BEAMchange
```

class BeamChangeCls

BeamChange commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:BEAMchange
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hesu.beamChange.fetch()
```

Queries the value of Beam Change field signaled in HE signal field for single user MIMO (HE-SIG-A) .

return

structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.3.2 BssColor

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:BSSColor
```

class BssColorCls

BssColor commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:BSSColor
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hesu.bssColor.fetch()
```

Queries the value of BSS Color field signaled in HE signal field for single user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.3.3 Bw

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:BW
```

class BwCls

Bw commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:BW
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hesu.bw.fetch()
```

Queries the value of Bandwidth field signaled in HE signal field for single user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.3.4 Coding

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:CODing
```

class CodingCls

Coding commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:CODing
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hesu.coding.fetch()
```

Queries the value of Coding field signaled in HE signal field for single user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.3.5 Crc

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:CRC
```

class CrcCls

Crc commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available
- Check: bool: Indicates passed or failed check verdict

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:CRC
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hesu.crc.fetch()
```

Queries the value of CRC field signaled in HE signal field for single user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.3.6 Dcm

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:DCM
```

class DcmCls

Dcm commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:DCM
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hesu.dcm.fetch()
```

Queries the value of DCM field signaled in HE signal field for single user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.3.7 Doppler

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:DOPPler
```

class DopplerCls

Doppler commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:DOPPler
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hesu.doppler.fetch()
```

Queries the value of Doppler field signaled in HE signal field for single user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.3.8 FormatPy

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:FORMat
```

class FormatPyCls

FormatPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:FORMat
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hesu.formatPy.fetch()
```

Queries the value of Format field signaled in HE signal field for single user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.3.9 GiltfSize

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:GILTfsize
```

class GiltfSizeCls

GiltfSize commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:GILTfsize
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hesu.giltfSize.fetch()
```

Queries the value of GI+LTF Size field signaled in HE signal field for single user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.3.10 Ldpc

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:LDPC
```

class LdpcCls

Ldpc commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:LDPC
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hesu.ldpc.fetch()
```

Queries the value of LDPC Extra Symbol Segment field signaled in HE signal field for single user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.3.11 Mcs

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:MCS
```

class McsCls

Mcs commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:MCS
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hesu.mcs.fetch()
```

Queries the value of MCS field signaled in HE signal field for single user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.3.12 Nsts

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:NSTS
```

class NstsCls

Nsts commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:NSTS
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hesu.nsts.fetch()
```

Queries the value of Nsts And Midamble Periodicity field signaled in the HE signal field for single user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.3.13 PeDisambiguity

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:PEDisambig
```

class PeDisambiguityCls

PeDisambiguity commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:PEDisambig
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hesu.peDisambiguity.fetch()
```

Queries the value of PE Disambiguity field signaled in HE signal field for single user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.3.14 PfecPadding

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:PFECpadding
```

class PfecPaddingCls

PfecPadding commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:PFECpadding
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hesu.pfecPadding.fetch()
```

Queries the value of Pre-FEC Padding Factor field signaled in HE signal field for single user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.3.15 Reserved<Reserved>

RepCap Settings

```
# Range: Nr1 .. Nr3
rc = driver.wlanMeas.multiEval.sinfo.hesu.reserved.repcap_reserved_get()
driver.wlanMeas.multiEval.sinfo.hesu.reserved.repcap_reserved_set(repcap.Reserved.Nr1)
```

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:REServed<index>
```

class ReservedCls

Reserved commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Reserved, default value after init: Reserved.Nr1

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch(*reserved=Reserved.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:REServed<index>
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hesu.reserved.
↪fetch(reserved = repcap.Reserved.Default)
```

Queries the value of Reserved field signaled in HE signal field for single user MIMO (HE-SIG-A) .

param reserved
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Reserved')

return
structure: for return value, see the help for FetchStruct structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.sinfo.hesu.reserved.clone()
```

6.4.1.6.3.16 SpatialReuse

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:SPATialreuse
```

class SpatialReuseCls

SpatialReuse commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:SPATialreuse
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hesu.spatialReuse.fetch()
```

Queries the value of Spatial Reuse field signaled in HE signal field for single user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.3.17 Stbc

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:STBC
```

class StbcCls

Stbc commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:STBC
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hesu.stbc.fetch()
```

Queries the value of STBC field signaled in HE signal field for single user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.3.18 Tail

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:TAIL
```

class TailCls

Tail commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available
- Check: bool: Indicates passed or failed check verdict

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:TAIL
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hesu.tail.fetch()
```

Queries the value of Tail field signaled in HE signal field for single user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.3.19 TxBf

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:TXBF
```

class TxBfCls

TxBf commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:TXBF
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hesu.txBf.fetch()
```

Queries the value of TxBF field signaled in HE signal field for single user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.3.20 TxOp

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:TXOP
```

class TxOpCls

TxOp commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:TXOP
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hesu.txOp.fetch()
```

Queries the value of TXOP field signaled in HE signal field for single user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.3.21 UlDl

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:ULDL
```

class UlDlCls

UlDl commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:ULDL
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hesu.ulDl.fetch()
```

Queries the value of UL/DL field signaled in HE signal field for single user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.4 Hetb

class HetbCls

Hetb commands group definition. 8 total commands, 8 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.sinfo.hetb.clone()
```

Subgroups

6.4.1.6.4.1 BssColor

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HETB:BSSColor
```

class BssColorCls

BssColor commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HETB:BSSColor
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hetb.bssColor.fetch()
```

Queries the value of BSS Color field signaled in HE signal field for trigger based uplink single user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.4.2 Bw

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HETB:BW
```

class BwCls

Bw commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HETB:BW
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hetb.bw.fetch()
```

Queries the value of Bandwidth field signaled in HE signal field for trigger based uplink single user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.4.3 Crc

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HETB:CRC
```

class CrcCls

Crc commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available
- Check: bool: Indicates passed or failed check verdict

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HETB:CRC
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hetb.crc.fetch()
```

Queries the value of CRC field signaled in HE signal field for trigger based uplink single user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.4.4 FormatPy

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HETB:FORMat
```

class FormatPyCls

FormatPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HETB:FORMat
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hetb.formatPy.fetch()
```

Queries the value of Format field signaled in HE signal field for trigger based uplink single user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.4.5 Reserved<Reserved>

RepCap Settings

```
# Range: Nr1 .. Nr3
rc = driver.wlanMeas.multiEval.sinfo.hetb.reserved.repcap_reserved_get()
driver.wlanMeas.multiEval.sinfo.hetb.reserved.repcap_reserved_set(repcap.Reserved.Nr1)
```

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HETB:REServed<index>
```

class ReservedCls

Reserved commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Reserved, default value after init: Reserved.Nr1

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch(*reserved=Reserved.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HETB:REServed<index>
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hetb.reserved.
↪fetch(reserved = repcap.Reserved.Default)
```

Queries the value of Reserved field signaled in HE signal field for trigger based uplink single user MIMO (HE-SIG-A) .

param reserved
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Reserved')

return
structure: for return value, see the help for FetchStruct structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.sinfo.hetb.reserved.clone()
```

6.4.1.6.4.6 SpatialReuse<Spatial>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.wlanMeas.multiEval.sinfo.hetb.spatialReuse.repcap_spatial_get()
driver.wlanMeas.multiEval.sinfo.hetb.spatialReuse.repcap_spatial_set(repcap.Spatial.Nr1)
```

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HETB:SPATialreuse<index>
```

class SpatialReuseCls

SpatialReuse commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Spatial, default value after init: Spatial.Nr1

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch(*spatial=Spatial.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HETB:SPATialreuse
↪<index>
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hetb.spatialReuse.
↪fetch(spatial = repcap.Spatial.Default)
```

Queries the value of Spatial Reuse field signaled in HE signal field for trigger based uplink single user MIMO (HE-SIG-A) .

param spatial
optional repeated capability selector. Default value: Nr1 (settable in the interface 'SpatialReuse')

return
structure: for return value, see the help for FetchStruct structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.sinfo.hetb.spatialReuse.clone()
```

6.4.1.6.4.7 Tail

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HETB:TAIL
```

class TailCls

Tail commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available
- Check: bool: Indicates passed or failed check verdict

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HETB:TAIL
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hetb.tail.fetch()
```

Queries the value of Tail field signaled in HE signal field for trigger based uplink single user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.4.8 TxOp

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HETB:TXOP
```

class TxOpCls

TxOp commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HETB:TXOP
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.hetb.txOp.fetch()
```

Queries the value of TXOP field signaled in HE signal field for trigger based uplink single user MIMO (HE-SIG-A) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.5 Htsig

class HtsigCls

Htsig commands group definition. 13 total commands, 13 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.sinfo.htsig.clone()
```

Subgroups

6.4.1.6.5.1 Aggregation

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HTSig:AGGRegation
```

class AggregationCls

Aggregation commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HTSig:AGGRegation
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.htsig.aggregation.fetch()
```

Queries the value of Aggregation field signaled in high throughput signal field for 802.11n signal (HT-SIG) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.5.2 Cbw

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HTSig:CBW
```

class CbwCls

Cbw commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HTSig:CBW
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.htsig.cbw.fetch()
```

Queries the value of CBW 20/40 field signaled in the high throughput signal field for 802.11n signal (HT-SIG) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.5.3 Crc

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HTSig:CRC
```

class CrcCls

Crc commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available
- Check: bool: Indicates passed or failed check verdict

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HTSig:CRC
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.htsig.crc.fetch()
```

Queries the value of CRC field signaled in high throughput signal field for 802.11n signal (HT-SIG) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.5.4 FecCoding

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HTSig:FECCoding
```

class FecCodingCls

FecCoding commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HTSig:FECCoding
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.htsig.fecCoding.fetch()
```

Queries the value of FEC Coding field signaled in high throughput signal field for 802.11n signal (HT-SIG) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.5.5 HtLength

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HTSig:HTLength
```

class HtLengthCls

HtLength commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HTSig:HTLength
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.htsig.htLength.fetch()
```

Queries the value of HT Length field signaled in high throughput signal field for 802.11n signal (HT-SIG) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.5.6 Mcs

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HTSig:MCS
```

class McsCls

Mcs commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HTSig:MCS
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.htsig.mcs.fetch()
```

Queries the value of the Modulation and Coding Scheme field signaled in the high throughput signal field for 802. 11n signal (HT-SIG) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.5.7 Ness

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HTSig:NESS
```

class NessCls

Ness commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HTSig:NESS
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.htsig.ness.fetch()
```

Queries the value of Number of Extension Spatial Streams field signaled in the high throughput signal field for 802. 11n signal (HT-SIG) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.5.8 Nsounding

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HTSig:NSOunding
```

class NsoundingCls

Nsounding commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HTSig:NSOunding
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.htsig.nsounding.fetch()
```

Queries the value of Not Sounding field signaled in the high throughput signal field for 802.11n signal (HT-SIG) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.5.9 Reserved

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HTSig:REServed
```

class ReservedCls

Reserved commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HTSig:REServed
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.htsig.reserved.fetch()
```

Queries the value of Reserved field signaled in high throughput signal field for 802.11n signal (HT-SIG) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.5.10 ShortGi

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HTSig:SHORtgi
```

class ShortGiCls

ShortGi commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HTSig:SHORtgi
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.htsig.shortGi.fetch()
```

Queries the value of Short GI field signaled in high throughput signal field for 802.11n signal (HT-SIG) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.5.11 Smoothing

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HTSig:SMOothing
```

class SmoothingCls

Smoothing commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HTSig:SMOothing
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.htsig.smoothing.fetch()
```

Queries the value of Smoothing field signaled in a high throughput signal field for 802.11n signal (HT-SIG) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.5.12 StbCoding

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HTSig:STBCoding
```

class StbCodingCls

StbCoding commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HTSig:STBCoding
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.htsig.stbCoding.fetch()
```

Queries the value of STBC field signaled in high throughput signal field for 802.11n signal (HT-SIG) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.5.13 Tail

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HTSig:TAIL
```

class TailCls

Tail commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available
- Check: bool: Indicates passed or failed check verdict

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HTSig:TAIL
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.htsig.tail.fetch()
```

Queries the value of Tail Bits field signaled in a high throughput signal field for 802.11n signal (HT-SIG) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.6 Lsig

class LsigCls

Lsig commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.sinfo.lsig.clone()
```

Subgroups

6.4.1.6.6.1 Length

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:LSIG:LENGth
```

class LengthCls

Length commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:LSIG:LENGth
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.lsig.length.fetch()
```

Queries the value of Length field signaled in legacy signal field for NON_HT signal (L-SIG) .

return

structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.6.2 Parity

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:LSIG:PARity
```

class ParityCls

Parity commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available
- Check: bool: Indicates passed or failed check verdict

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:LSIG:PARity
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.lsig.parity.fetch()
```

Queries the value of the Parity field signaled in legacy signal field for NON_HT signal (L-SIG) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.6.3 Rate

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:LSIG:RATE
```

class RateCls

Rate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:LSIG:RATE
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.lsig.rate.fetch()
```

Queries the value of Rate field signaled in legacy signal field for NON_HT signal (L-SIG) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.6.4 Reserved

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:LSIG:REServed
```

class ReservedCls

Reserved commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:LSIG:REServed
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.lsig.reserved.fetch()
```

Queries the value of Reserved field signaled in legacy signal field for NON_HT signal (L-SIG) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.6.5 Tail

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:LSIG:TAIL
```

class TailCls

Tail commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available
- Check: bool: Indicates passed or failed check verdict

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:LSIG:TAIL
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.lsig.tail.fetch()
```

Queries the value of Tail field signaled in legacy signal field for NON_HT signal (L-SIG) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.7 VhtSig

class VhtSigCls

VhtSig commands group definition. 15 total commands, 15 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.sinfo.vhtSig.clone()
```

Subgroups

6.4.1.6.7.1 Beamformed

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:VHTSig:BEAMformed
```

class BeamformedCls

Beamformed commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:VHTSig:BEAMformed
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.vhtSig.beamformed.fetch()
```

Queries the value of Beamformed field signaled in very high throughput signal field for 802.11ac signal (VHT-SIG) .

return

structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.7.2 Bw

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:VHTSig:BW
```

class BwCls

Bw commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:VHTSig:BW
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.vhtSig.bw.fetch()
```

Queries the value of BW field signaled in very high throughput signal field for 802.11ac signal (VHT-SIG) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.7.3 Crc

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:VHTSig:CRC
```

class CrcCls

Crc commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available
- Check: bool: Indicates passed or failed check verdict

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:VHTSig:CRC
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.vhtSig.crc.fetch()
```

Queries the value of CRC field signaled in very high throughput signal field for 802.11ac signal (VHT-SIG) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.7.4 FecCoding

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:VHTSig:FECCoding
```

class FecCodingCls

FecCoding commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:VHTSig:FECCoding
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.vhtSig.fecCoding.fetch()
```

Queries the value of SU/MU[0] Coding field signaled in very high throughput signal field for 802.11ac signal (VHT-SIG) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.7.5 Gid

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:VHTSig:GID
```

class GidCls

Gid commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:VHTSig:GID
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.vhtSig.gid.fetch()
```

Queries the value of the Group ID field signaled in a very high throughput signal field for 802.11ac signal (VHT-SIG) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.7.6 Ldpc

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:VHTSig:LDPC
```

class LdpcCls

Ldpc commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:VHTSig:LDPC
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.vhtSig.ldpc.fetch()
```

Queries the value of LDPC Extra OFDM Symbol field signaled in a very high throughput signal field for 802.11ac signal (VHT-SIG) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.7.7 Paid

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:VHTSig:PAID
```

class PaidCls

Paid commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:VHTSig:PAID
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.vhtSig.paid.fetch()
```

Queries the value of Partial AID field signaled in very high throughput signal field for 802.11ac signal (VHT-SIG) . For the filed NSTS refer to: method RsCMPX_WlanMeas.WlanMeas.MultiEval.Sinfo.VhtSig.Sunsts.fetch

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.7.8 Reserved<Reserved>

RepCap Settings

```
# Range: Nr1 .. Nr3
rc = driver.wlanMeas.multiEval.sinfo.vhtSig.reserved.repcap_reserved_get()
driver.wlanMeas.multiEval.sinfo.vhtSig.reserved.repcap_reserved_set(repcap.Reserved.Nr1)
```

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:VHTSig:REServed<index>
```

class ReservedCls

Reserved commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Reserved, default value after init: Reserved.Nr1

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch(*reserved=Reserved.Default*) → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:VHTSig:REServed
↪<index>
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.vhtSig.reserved.
↪fetch(reserved = repcap.Reserved.Default)
```

Queries the value of Reserved field signaled in very high throughput signal field for 802.11ac signal (VHT-SIG) .

param reserved
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Reserved')

return
structure: for return value, see the help for FetchStruct structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.sinfo.vhtSig.reserved.clone()
```

6.4.1.6.7.9 Sdisambiguity

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:VHTSig:SDISambig
```

class SdisambiguityCls

Sdisambiguity commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:VHTSig:SDISambig
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.vhtSig.sdisambiguity.
↪fetch()
```

Queries the value of Short GI NSYM Disambiguation field signaled in a very high throughput signal field for 802. 11ac signal (VHT-SIG) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.7.10 Sgi

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:VHTSig:SGI
```

class SgiCls

Sgi commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:VHTSig:SGI
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.vhtSig.sgi.fetch()
```

Queries the value of Short GI field signaled in very high throughput signal field for 802.11ac signal (VHT-SIG) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.7.11 Smcs

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:VHTSig:SMCS
```

class SmcsCls

Smcs commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:VHTSig:SMCS
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.vhtSig.smcs.fetch()
```

Queries the value of SU VHT-MCS/ MU[1- 3] Coding field signaled in very high throughput signal field for 802.11ac signal (VHT-SIG) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.7.12 Stbc

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:VHTSig:STBC
```

class StbcCls

Stbc commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available

- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:VHTSig:STBC
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.vhtSig.stbc.fetch()
```

Queries the value of STBC field signaled in very high throughput signal field for 802.11ac signal (VHT-SIG) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.7.13 Sunsts

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:VHTSig:SUNSts
```

class SunstsCls

Sunsts commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: float: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:VHTSig:SUNSts
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.vhtSig.sunsts.fetch()
```

Queries the value of NSTS field signaled in very high throughput signal field for 802.11ac signal (VHT-SIG) . For the filed Partial AID refer to: method RsCMPX_WlanMeas.WlanMeas.MultiEval.Sinfo.VhtSig.Paid.fetch

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.7.14 Tail

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:VHTSig:TAIL
```

class TailCls

Tail commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available
- Check: bool: Indicates passed or failed check verdict

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:VHTSig:TAIL
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.vhtSig.tail.fetch()
```

Queries the value of Tail field signaled in very high throughput signal field for 802.11ac signal (VHT-SIG) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.6.7.15 TxOp

SCPI Command :

```
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:VHTSig:TXOP
```

class TxOpCls

TxOp commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Value_Bin: str: No parameter help available
- Value_Dec: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:VHTSig:TXOP
value: FetchStruct = driver.wlanMeas.multiEval.sinfo.vhtSig.txOp.fetch()
```

Queries the value of **TXOP_PS_** NOT_ALLOWED field signaled in very high throughput signal field for 802. 11ac signal (VHT-SIG) .

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.7 SpectrFlatness

class SpectrFlatnessCls

SpectrFlatness commands group definition. 40 total commands, 6 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.spectrFlatness.clone()
```

Subgroups

6.4.1.7.1 Average

SCPI Commands :

```
READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:AVERage
FETCh:WLAN:MEASurement<instance>:MEValuation:SFLatness:AVERage
CALCulate:WLAN:MEASurement<instance>:MEValuation:SFLatness:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate() → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<instance>:MEValuation:SFLatness:AVERage
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.spectrFlatness.
↪average.calculate()
```

Returns the margin values of the spectrum flatness measurement for the current, average, minimum and maximum traces. A positive margin indicates a violation of the spectrum flatness limit. The respective trace value is located above the upper or below the lower limit line. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

return

margins: Up to 10 comma-separated list of margins (one value per subcarrier range from left to right) Value 1: trace margin to the upper spectrum flatness limit For bandwidths 80 MHz, the margin is only relevant for the left 80 MHz segment (segment 1) . Value 2 to 5: trace margins to the lower spectrum flatness limit For bandwidths 80 MHz, the margins are only relevant for the left 80 MHz segment (segment 1) . Value 6: trace margin to the upper spectrum flatness limit for the right 80 MHz segment (segment 2) . This margin is only relevant for bandwidths 80 MHz. Value 7 to 10: trace margins to the lower spectrum flatness limit for the right 80 MHz segment (segment 2) . Values 6 to 10 are only relevant for bandwidths 80 MHz.

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SFLatness:AVERage
value: List[float] = driver.wlanMeas.multiEval.spectrFlatness.average.fetch()
```

Returns the margin values of the spectrum flatness measurement for the current, average, minimum and maximum traces. A positive margin indicates a violation of the spectrum flatness limit. The respective trace value is located above the upper or below the lower limit line. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

return
margins: Up to 10 comma-separated list of margins (one value per subcarrier range from left to right) Value 1: trace margin to the upper spectrum flatness limit For bandwidths 80 MHz, the margin is only relevant for the left 80 MHz segment (segment 1) . Value 2 to 5: trace margins to the lower spectrum flatness limit For bandwidths 80 MHz, the margins are only relevant for the left 80 MHz segment (segment 1) . Value 6: trace margin to the upper spectrum flatness limit for the right 80 MHz segment (segment 2) . This margin is only relevant for bandwidths 80 MHz. Value 7 to 10: trace margins to the lower spectrum flatness limit for the right 80 MHz segment (segment 2) . Values 6 to 10 are only relevant for bandwidths 80 MHz.

read() → List[float]

```
# SCPI: READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:AVERage
value: List[float] = driver.wlanMeas.multiEval.spectrFlatness.average.read()
```

Returns the margin values of the spectrum flatness measurement for the current, average, minimum and maximum traces. A positive margin indicates a violation of the spectrum flatness limit. The respective trace value is located above the upper or below the lower limit line. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

return
margins: Up to 10 comma-separated list of margins (one value per subcarrier range from left to right) Value 1: trace margin to the upper spectrum flatness limit For bandwidths 80 MHz, the margin is only relevant for the left 80 MHz segment (segment 1) . Value 2 to 5: trace margins to the lower spectrum flatness limit For bandwidths 80 MHz, the margins are only relevant for the left 80 MHz segment (segment 1) . Value 6: trace margin to the upper spectrum flatness limit for the right 80 MHz segment (segment 2) . This margin is only relevant for bandwidths 80 MHz. Value 7 to 10: trace margins to the lower spectrum flatness limit for the right 80 MHz segment (segment 2) . Values 6 to 10 are only relevant for bandwidths 80 MHz.

6.4.1.7.2 Current

SCPI Commands :

```
READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:CURRent
FETCh:WLAN:MEASurement<instance>:MEValuation:SFLatness:CURRent
CALCulate:WLAN:MEASurement<instance>:MEValuation:SFLatness:CURRent
```

class CurrentCls

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate() → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<instance>:MEValuation:SFLatness:CURRent
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.spectrFlatness.
↪current.calculate()
```

Returns the margin values of the spectrum flatness measurement for the current, average, minimum and maximum traces. A positive margin indicates a violation of the spectrum flatness limit. The respective trace value is located above the upper or below the lower limit line. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

return
margins: Up to 10 comma-separated list of margins (one value per subcarrier range from left to right) Value 1: trace margin to the upper spectrum flatness limit For bandwidths 80 MHz, the margin is only relevant for the left 80 MHz segment (segment 1) . Value 2 to 5: trace margins to the lower spectrum flatness limit For bandwidths 80 MHz, the margins are only relevant for the left 80 MHz segment (segment 1) . Value 6: trace margin to the upper spectrum flatness limit for the right 80 MHz segment (segment 2) . This margin is only relevant for bandwidths 80 MHz. Value 7 to 10: trace margins to the lower spectrum flatness limit for the right 80 MHz segment (segment 2) . Values 6 to 10 are only relevant for bandwidths 80 MHz.

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SFLatness:CURRent
value: List[float] = driver.wlanMeas.multiEval.spectrFlatness.current.fetch()
```

Returns the margin values of the spectrum flatness measurement for the current, average, minimum and maximum traces. A positive margin indicates a violation of the spectrum flatness limit. The respective trace value is located above the upper or below the lower limit line. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

return
margins: Up to 10 comma-separated list of margins (one value per subcarrier range from left to right) Value 1: trace margin to the upper spectrum flatness limit For bandwidths 80 MHz, the margin is only relevant for the left 80 MHz segment (segment 1) . Value 2 to 5: trace margins to the lower spectrum flatness limit For bandwidths 80 MHz, the margins are only relevant for the left 80 MHz segment (segment 1) . Value 6: trace margin to the upper spectrum flatness limit for the right 80 MHz segment (segment 2) . This margin is only relevant for bandwidths 80 MHz. Value 7 to 10: trace margins to the lower spectrum flatness limit for the right 80 MHz segment (segment 2) . Values 6 to 10 are only relevant for bandwidths 80 MHz.

read() → List[float]

```
# SCPI: READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:CURRent
value: List[float] = driver.wlanMeas.multiEval.spectrFlatness.current.read()
```

Returns the margin values of the spectrum flatness measurement for the current, average, minimum and maximum traces. A positive margin indicates a violation of the spectrum flatness limit. The respective trace value is located above the upper or below the lower limit line. The values described below are returned

by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

return
margins: Up to 10 comma-separated list of margins (one value per subcarrier range from left to right) Value 1: trace margin to the upper spectrum flatness limit For bandwidths 80 MHz, the margin is only relevant for the left 80 MHz segment (segment 1) . Value 2 to 5: trace margins to the lower spectrum flatness limit For bandwidths 80 MHz, the margins are only relevant for the left 80 MHz segment (segment 1) . Value 6: trace margin to the upper spectrum flatness limit for the right 80 MHz segment (segment 2) . This margin is only relevant for bandwidths 80 MHz. Value 7 to 10: trace margins to the lower spectrum flatness limit for the right 80 MHz segment (segment 2) . Values 6 to 10 are only relevant for bandwidths 80 MHz.

6.4.1.7.3 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:MAXimum
FETCh:WLAN:MEASurement<instance>:MEValuation:SFLatness:MAXimum
CALCulate:WLAN:MEASurement<instance>:MEValuation:SFLatness:MAXimum
```

class MaximumCls

Maximum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate() → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<instance>:MEValuation:SFLatness:MAXimum
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.spectrFlatness.
↪maximum.calculate()
```

Returns the margin values of the spectrum flatness measurement for the current, average, minimum and maximum traces. A positive margin indicates a violation of the spectrum flatness limit. The respective trace value is located above the upper or below the lower limit line. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

return
margins: Up to 10 comma-separated list of margins (one value per subcarrier range from left to right) Value 1: trace margin to the upper spectrum flatness limit For bandwidths 80 MHz, the margin is only relevant for the left 80 MHz segment (segment 1) . Value 2 to 5: trace margins to the lower spectrum flatness limit For bandwidths 80 MHz, the margins are only relevant for the left 80 MHz segment (segment 1) . Value 6: trace margin to the upper spectrum flatness limit for the right 80 MHz segment (segment 2) . This margin is only relevant for bandwidths 80 MHz. Value 7 to 10: trace margins to the lower spectrum flatness limit for the right 80 MHz segment (segment 2) . Values 6 to 10 are only relevant for bandwidths 80 MHz.

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SFLatness:MAXimum
value: List[float] = driver.wlanMeas.multiEval.spectrFlatness.maximum.fetch()
```

Returns the margin values of the spectrum flatness measurement for the current, average, minimum and maximum traces. A positive margin indicates a violation of the spectrum flatness limit. The respective trace value is located above the upper or below the lower limit line. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

return
margins: Up to 10 comma-separated list of margins (one value per subcarrier range from left to right) Value 1: trace margin to the upper spectrum flatness limit For bandwidths 80 MHz, the margin is only relevant for the left 80 MHz segment (segment 1) . Value 2 to 5: trace margins to the lower spectrum flatness limit For bandwidths 80 MHz, the margins are only relevant for the left 80 MHz segment (segment 1) . Value 6: trace margin to the upper spectrum flatness limit for the right 80 MHz segment (segment 2) . This margin is only relevant for bandwidths 80 MHz. Value 7 to 10: trace margins to the lower spectrum flatness limit for the right 80 MHz segment (segment 2) . Values 6 to 10 are only relevant for bandwidths 80 MHz.

read() → List[float]

```
# SCPI: READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:MAXimum
value: List[float] = driver.wlanMeas.multiEval.spectrFlatness.maximum.read()
```

Returns the margin values of the spectrum flatness measurement for the current, average, minimum and maximum traces. A positive margin indicates a violation of the spectrum flatness limit. The respective trace value is located above the upper or below the lower limit line. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

return
margins: Up to 10 comma-separated list of margins (one value per subcarrier range from left to right) Value 1: trace margin to the upper spectrum flatness limit For bandwidths 80 MHz, the margin is only relevant for the left 80 MHz segment (segment 1) . Value 2 to 5: trace margins to the lower spectrum flatness limit For bandwidths 80 MHz, the margins are only relevant for the left 80 MHz segment (segment 1) . Value 6: trace margin to the upper spectrum flatness limit for the right 80 MHz segment (segment 2) . This margin is only relevant for bandwidths 80 MHz. Value 7 to 10: trace margins to the lower spectrum flatness limit for the right 80 MHz segment (segment 2) . Values 6 to 10 are only relevant for bandwidths 80 MHz.

6.4.1.7.4 Mimo<Mimo>

RepCap Settings

```
# Range: Nr1 .. Nr8
rc = driver.wlanMeas.multiEval.spectrFlatness.mimo.repcap_mimo_get()
driver.wlanMeas.multiEval.spectrFlatness.mimo.repcap_mimo_set(repcap.Mimo.Nr1)
```

class MimoCls

Mimo commands group definition. 20 total commands, 5 Subgroups, 0 group commands Repeated Capability: Mimo, default value after init: Mimo.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.spectrFlatness.mimo.clone()
```

Subgroups

6.4.1.7.4.1 Average

SCPI Commands :

```
READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:MIMO<n>:AVERage
FETCh:WLAN:MEASurement<instance>:MEValuation:SFLatness:MIMO<n>:AVERage
CALCulate:WLAN:MEASurement<instance>:MEValuation:SFLatness:MIMO<n>:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*mimo=Mimo.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<instance>:MEValuation:SFLatness:MIMO<n>
↪:AVERage
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.spectrFlatness.
↪mimo.average.calculate(mimo = repcap.Mimo.Default)
```

Return the single value margins for true MIMO measurements, antenna/stream number <n>. There are current, average, minimum, and maximum results. A positive margin indicates a violation of the spectrum flatness limit. The respective trace value is located above the upper or below the lower limit line. For the queries of subcarrier indices for spectrum flatness margins, see: method RsCMPX_WlanMeas.WlanMeas.MultiEval.SpectrFlatness.Mimo.X.Current.fetch etc. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
spec_flat_margins_tx: Up to 10 comma-separated list of margins (one value per subcarrier range from left to right) Value 1: trace margin to the upper spectrum flatness limit For bandwidths 80 MHz, the margin is only relevant for the left 80 MHz segment (segment 1) . Value 2 to 5: trace margins to the lower spectrum flatness limit For bandwidths 80 MHz, the margins are only relevant for the left 80 MHz segment (segment 1) . Value 6: trace margin to the upper spectrum flatness limit for the right 80 MHz segment (segment 2) . This margin is only relevant for bandwidths 80 MHz. Value 7 to 10: trace margins to the lower spectrum flatness limit for the right 80 MHz segment (segment 2) . Values 6 to 10 are only relevant for bandwidths 80 MHz.

fetch(*mimo=Mimo.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SFLatness:MIMO<n>:AVERage
value: List[float] = driver.wlanMeas.multiEval.spectrFlatness.mimo.average.
↪fetch(mimo = repcap.Mimo.Default)
```

Return the single value margins for true MIMO measurements, antenna/stream number <n>. There are current, average, minimum, and maximum results. A positive margin indicates a violation of the spectrum flatness limit. The respective trace value is located above the upper or below the lower limit line. For the queries of subcarrier indices for spectrum flatness margins, see: method RsCMPX_WlanMeas.WlanMeas.MultiEval.SpectrFlatness.Mimo.X.Current.fetch etc. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
spec_flat_margins_tx: Up to 10 comma-separated list of margins (one value per subcarrier range from left to right) Value 1: trace margin to the upper spectrum flatness limit For bandwidths 80 MHz, the margin is only relevant for the left 80 MHz segment (segment 1) . Value 2 to 5: trace margins to the lower spectrum flatness limit For bandwidths 80 MHz, the margins are only relevant for the left 80 MHz segment (segment 1) . Value 6: trace margin to the upper spectrum flatness limit for the right 80 MHz segment (segment 2) . This margin is only relevant for bandwidths 80 MHz. Value 7 to 10: trace margins to the lower spectrum flatness limit for the right 80 MHz segment (segment 2) . Values 6 to 10 are only relevant for bandwidths 80 MHz.

read(*mimo=Mimo.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:MIMO<n>:AVERage
value: List[float] = driver.wlanMeas.multiEval.spectrFlatness.mimo.average.
↪read(mimo = repcap.Mimo.Default)
```

Return the single value margins for true MIMO measurements, antenna/stream number <n>. There are current, average, minimum, and maximum results. A positive margin indicates a violation of the spectrum flatness limit. The respective trace value is located above the upper or below the lower limit line. For the queries of subcarrier indices for spectrum flatness margins, see: method RsCMPX_WlanMeas.WlanMeas.MultiEval.SpectrFlatness.Mimo.X.Current.fetch etc. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
spec_flat_margins_tx: Up to 10 comma-separated list of margins (one value per subcarrier range from left to right) Value 1: trace margin to the upper spectrum flatness limit For bandwidths 80 MHz, the margin is only relevant for the left 80 MHz segment (segment 1) . Value 2 to 5: trace margins to the lower spectrum flatness limit For bandwidths 80 MHz, the margins are only relevant for the left 80 MHz segment (segment 1) . Value 6: trace margin to the upper spectrum flatness limit for the right 80 MHz segment (segment 2) . This margin is only relevant for bandwidths 80 MHz. Value 7 to 10: trace margins to the lower spectrum flatness limit for the right 80 MHz segment (segment 2) . Values 6 to 10 are only relevant for bandwidths 80 MHz.

6.4.1.7.4.2 Current

SCPI Commands :

```
READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:MIMO<n>:CURRent
FETCh:WLAN:MEASurement<instance>:MEValuation:SFLatness:MIMO<n>:CURRent
CALCulate:WLAN:MEASurement<instance>:MEValuation:SFLatness:MIMO<n>:CURRent
```

`class CurrentCls`

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

`calculate`(*mimo=Mimo.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<instance>:MEValuation:SFLatness:MIMO<n>
↪:CURRent
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.spectrFlatness.
↪mimo.current.calculate(mimo = repcap.Mimo.Default)
```

Return the single value margins for true MIMO measurements, antenna/stream number <n>. There are current, average, minimum, and maximum results. A positive margin indicates a violation of the spectrum flatness limit. The respective trace value is located above the upper or below the lower limit line. For the queries of subcarrier indices for spectrum flatness margins, see: method RsCMPX_WlanMeas.WlanMeas.MultiEval.SpectrFlatness.Mimo.X.Current.fetch etc. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
spec_flat_margins_tx: Up to 10 comma-separated list of margins (one value per subcarrier range from left to right) Value 1: trace margin to the upper spectrum flatness limit For bandwidths 80 MHz, the margin is only relevant for the left 80 MHz segment (segment 1) . Value 2 to 5: trace margins to the lower spectrum flatness limit For bandwidths 80 MHz, the margins are only relevant for the left 80 MHz segment (segment 1) . Value 6: trace margin to the upper spectrum flatness limit for the right 80 MHz segment (segment 2) . This margin is only relevant for bandwidths 80 MHz. Value 7

to 10: trace margins to the lower spectrum flatness limit for the right 80 MHz segment (segment 2) . Values 6 to 10 are only relevant for bandwidths 80 MHz.

fetch(*mimo=Mimo.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SFLatness:MIMO<n>:CURRent
value: List[float] = driver.wlanMeas.multiEval.spectrFlatness.mimo.current.
↪fetch(mimo = repcap.Mimo.Default)
```

Return the single value margins for true MIMO measurements, antenna/stream number <n>. There are current, average, minimum, and maximum results. A positive margin indicates a violation of the spectrum flatness limit. The respective trace value is located above the upper or below the lower limit line. For the queries of subcarrier indices for spectrum flatness margins, see: method RsCMPX_WlanMeas.WlanMeas.MultiEval.SpectrFlatness.Mimo.X.Current.fetch etc. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
spec_flat_margins_tx: Up to 10 comma-separated list of margins (one value per subcarrier range from left to right) Value 1: trace margin to the upper spectrum flatness limit For bandwidths 80 MHz, the margin is only relevant for the left 80 MHz segment (segment 1) . Value 2 to 5: trace margins to the lower spectrum flatness limit For bandwidths 80 MHz, the margins are only relevant for the left 80 MHz segment (segment 1) . Value 6: trace margin to the upper spectrum flatness limit for the right 80 MHz segment (segment 2) . This margin is only relevant for bandwidths 80 MHz. Value 7 to 10: trace margins to the lower spectrum flatness limit for the right 80 MHz segment (segment 2) . Values 6 to 10 are only relevant for bandwidths 80 MHz.

read(*mimo=Mimo.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:MIMO<n>:CURRent
value: List[float] = driver.wlanMeas.multiEval.spectrFlatness.mimo.current.
↪read(mimo = repcap.Mimo.Default)
```

Return the single value margins for true MIMO measurements, antenna/stream number <n>. There are current, average, minimum, and maximum results. A positive margin indicates a violation of the spectrum flatness limit. The respective trace value is located above the upper or below the lower limit line. For the queries of subcarrier indices for spectrum flatness margins, see: method RsCMPX_WlanMeas.WlanMeas.MultiEval.SpectrFlatness.Mimo.X.Current.fetch etc. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
spec_flat_margins_tx: Up to 10 comma-separated list of margins (one value per subcarrier range from left to right) Value 1: trace margin to the upper spectrum flatness limit For bandwidths 80 MHz, the margin is only relevant for the left 80 MHz segment

(segment 1) . Value 2 to 5: trace margins to the lower spectrum flatness limit For bandwidths 80 MHz, the margins are only relevant for the left 80 MHz segment (segment 1) . Value 6: trace margin to the upper spectrum flatness limit for the right 80 MHz segment (segment 2) . This margin is only relevant for bandwidths 80 MHz. Value 7 to 10: trace margins to the lower spectrum flatness limit for the right 80 MHz segment (segment 2) . Values 6 to 10 are only relevant for bandwidths 80 MHz.

6.4.1.7.4.3 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:MIMO<n>:MAXimum
FETCh:WLAN:MEASurement<instance>:MEValuation:SFLatness:MIMO<n>:MAXimum
CALCulate:WLAN:MEASurement<instance>:MEValuation:SFLatness:MIMO<n>:MAXimum
```

class MaximumCls

Maximum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*mimo=Mimo.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<instance>:MEValuation:SFLatness:MIMO<n>
↪:MAXimum
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.spectrFlatness.
↪mimo.maximum.calculate(mimo = repcap.Mimo.Default)
```

Return the single value margins for true MIMO measurements, antenna/stream number <n>. There are current, average, minimum, and maximum results. A positive margin indicates a violation of the spectrum flatness limit. The respective trace value is located above the upper or below the lower limit line. For the queries of subcarrier indices for spectrum flatness margins, see: method RsCMPX_WlanMeas.WlanMeas.MultiEval.SpectrFlatness.Mimo.X.Current.fetch etc. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
spec_flat_margins_tx: Up to 10 comma-separated list of margins (one value per subcarrier range from left to right) Value 1: trace margin to the upper spectrum flatness limit For bandwidths 80 MHz, the margin is only relevant for the left 80 MHz segment (segment 1) . Value 2 to 5: trace margins to the lower spectrum flatness limit For bandwidths 80 MHz, the margins are only relevant for the left 80 MHz segment (segment 1) . Value 6: trace margin to the upper spectrum flatness limit for the right 80 MHz segment (segment 2) . This margin is only relevant for bandwidths 80 MHz. Value 7 to 10: trace margins to the lower spectrum flatness limit for the right 80 MHz segment (segment 2) . Values 6 to 10 are only relevant for bandwidths 80 MHz.

fetch(*mimo=Mimo.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SFLatness:MIMO<n>:MAXimum
value: List[float] = driver.wlanMeas.multiEval.spectrFlatness.mimo.maximum.
↪fetch(mimo = repcap.Mimo.Default)
```

Return the single value margins for true MIMO measurements, antenna/stream number <n>. There are current, average, minimum, and maximum results. A positive margin indicates a violation of the spectrum flatness limit. The respective trace value is located above the upper or below the lower limit line. For the queries of subcarrier indices for spectrum flatness margins, see: method RsCMPX_WlanMeas.WlanMeas.MultiEval.SpectrFlatness.Mimo.X.Current.fetch etc. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
spec_flat_margins_tx: Up to 10 comma-separated list of margins (one value per subcarrier range from left to right) Value 1: trace margin to the upper spectrum flatness limit For bandwidths 80 MHz, the margin is only relevant for the left 80 MHz segment (segment 1) . Value 2 to 5: trace margins to the lower spectrum flatness limit For bandwidths 80 MHz, the margins are only relevant for the left 80 MHz segment (segment 1) . Value 6: trace margin to the upper spectrum flatness limit for the right 80 MHz segment (segment 2) . This margin is only relevant for bandwidths 80 MHz. Value 7 to 10: trace margins to the lower spectrum flatness limit for the right 80 MHz segment (segment 2) . Values 6 to 10 are only relevant for bandwidths 80 MHz.

read(*mimo=Mimo.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:MIMO<n>:MAXimum
value: List[float] = driver.wlanMeas.multiEval.spectrFlatness.mimo.maximum.
↪read(mimo = repcap.Mimo.Default)
```

Return the single value margins for true MIMO measurements, antenna/stream number <n>. There are current, average, minimum, and maximum results. A positive margin indicates a violation of the spectrum flatness limit. The respective trace value is located above the upper or below the lower limit line. For the queries of subcarrier indices for spectrum flatness margins, see: method RsCMPX_WlanMeas.WlanMeas.MultiEval.SpectrFlatness.Mimo.X.Current.fetch etc. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
spec_flat_margins_tx: Up to 10 comma-separated list of margins (one value per subcarrier range from left to right) Value 1: trace margin to the upper spectrum flatness limit For bandwidths 80 MHz, the margin is only relevant for the left 80 MHz segment (segment 1) . Value 2 to 5: trace margins to the lower spectrum flatness limit For bandwidths 80 MHz, the margins are only relevant for the left 80 MHz segment (segment 1) . Value 6: trace margin to the upper spectrum flatness limit for the right 80 MHz segment (segment 2) . This margin is only relevant for bandwidths 80 MHz. Value 7 to 10: trace margins to the lower spectrum flatness limit for the right 80 MHz segment (segment 2) . Values 6 to 10 are only relevant for bandwidths 80 MHz.

6.4.1.7.4.4 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:MIMO<n>:MINimum
FETCh:WLAN:MEASurement<instance>:MEValuation:SFLatness:MIMO<n>:MINimum
CALCulate:WLAN:MEASurement<instance>:MEValuation:SFLatness:MIMO<n>:MINimum
```

class MinimumCls

Minimum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*mimo=Mimo.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<instance>:MEValuation:SFLatness:MIMO<n>
↪:MINimum
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.spectrFlatness.
↪mimo.minimum.calculate(mimo = repcap.Mimo.Default)
```

Return the single value margins for true MIMO measurements, antenna/stream number <n>. There are current, average, minimum, and maximum results. A positive margin indicates a violation of the spectrum flatness limit. The respective trace value is located above the upper or below the lower limit line. For the queries of subcarrier indices for spectrum flatness margins, see: method RsCMPX_WlanMeas.WlanMeas.MultiEval.SpectrFlatness.Mimo.X.Current.fetch etc. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
spec_flat_margins_tx: Up to 10 comma-separated list of margins (one value per subcarrier range from left to right) Value 1: trace margin to the upper spectrum flatness limit For bandwidths 80 MHz, the margin is only relevant for the left 80 MHz segment (segment 1) . Value 2 to 5: trace margins to the lower spectrum flatness limit For bandwidths 80 MHz, the margins are only relevant for the left 80 MHz segment (segment 1) . Value 6: trace margin to the upper spectrum flatness limit for the right 80 MHz segment (segment 2) . This margin is only relevant for bandwidths 80 MHz. Value 7 to 10: trace margins to the lower spectrum flatness limit for the right 80 MHz segment (segment 2) . Values 6 to 10 are only relevant for bandwidths 80 MHz.

fetch(*mimo=Mimo.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SFLatness:MIMO<n>:MINimum
value: List[float] = driver.wlanMeas.multiEval.spectrFlatness.mimo.minimum.
↪fetch(mimo = repcap.Mimo.Default)
```

Return the single value margins for true MIMO measurements, antenna/stream number <n>. There are current, average, minimum, and maximum results. A positive margin indicates a violation of the spectrum flatness limit. The respective trace value is located above the upper or below the lower limit line. For the queries of subcarrier indices for spectrum flatness margins, see: method RsCMPX_WlanMeas.WlanMeas.MultiEval.SpectrFlatness.Mimo.X.Current.fetch etc. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
spec_flat_margins_tx: Up to 10 comma-separated list of margins (one value per subcarrier range from left to right) Value 1: trace margin to the upper spectrum flatness limit For bandwidths 80 MHz, the margin is only relevant for the left 80 MHz segment (segment 1) . Value 2 to 5: trace margins to the lower spectrum flatness limit For bandwidths 80 MHz, the margins are only relevant for the left 80 MHz segment (segment 1) . Value 6: trace margin to the upper spectrum flatness limit for the right 80 MHz segment (segment 2) . This margin is only relevant for bandwidths 80 MHz. Value 7 to 10: trace margins to the lower spectrum flatness limit for the right 80 MHz segment (segment 2) . Values 6 to 10 are only relevant for bandwidths 80 MHz.

read(*mimo=Mimo.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:MIMO<n>:MINimum
value: List[float] = driver.wlanMeas.multiEval.spectrFlatness.mimo.minimum.
↪read(mimo = repcap.Mimo.Default)
```

Return the single value margins for true MIMO measurements, antenna/stream number <n>. There are current, average, minimum, and maximum results. A positive margin indicates a violation of the spectrum flatness limit. The respective trace value is located above the upper or below the lower limit line. For the queries of subcarrier indices for spectrum flatness margins, see: method RsCMPX_WlanMeas.WlanMeas.MultiEval.SpectrFlatness.Mimo.X.Current.fetch etc. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
spec_flat_margins_tx: Up to 10 comma-separated list of margins (one value per subcarrier range from left to right) Value 1: trace margin to the upper spectrum flatness limit For bandwidths 80 MHz, the margin is only relevant for the left 80 MHz segment (segment 1) . Value 2 to 5: trace margins to the lower spectrum flatness limit For bandwidths 80 MHz, the margins are only relevant for the left 80 MHz segment (segment 1) . Value 6: trace margin to the upper spectrum flatness limit for the right 80 MHz segment (segment 2) . This margin is only relevant for bandwidths 80 MHz. Value 7 to 10: trace margins to the lower spectrum flatness limit for the right 80 MHz segment (segment 2) . Values 6 to 10 are only relevant for bandwidths 80 MHz.

6.4.1.7.4.5 X

class XCls

X commands group definition. 8 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.spectrFlatness.mimo.x.clone()
```

Subgroups

6.4.1.7.4.6 Average

SCPI Commands :

```
READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:MIMO<n>:X:AVERage
FETCh:WLAN:MEASurement<instance>:MEValuation:SFLatness:MIMO<n>:X:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*mimo=Mimo.Default*) → List[int]

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SFLatness:MIMO<n>:X:AVERage
value: List[int] = driver.wlanMeas.multiEval.spectrFlatness.mimo.x.average.
↪fetch(mimo = repcap.Mimo.Default)
```

Return the subcarrier indices (x positions of the worst values) for the current, average, minimum and maximum margin values for true MIMO, antenna/stream number <n>. For the queries of spectrum flatness margins, see: method RsCMPX_WlanMeas.WlanMeas.MultiEval.SpectrFlatness.Mimo.Current.fetch etc.

Suppressed linked return values: reliability

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
spec_flat_margins_segments_tx: Up to 10 comma-separated values of subcarrier indices (one index per subcarrier range from left to right) Value 1: subcarrier index of the trace margin to the upper spectrum flatness limit For bandwidths 80 MHz, the value is only relevant for the left 80 MHz segment (segment 1) . Value 2 to 5: subcarrier indices, for the trace margin to the lower spectrum flatness limit For bandwidths 80 MHz, the values are only relevant for the left 80 MHz segment (segment 1) . Value 6: subcarrier index of the trace margin to the upper spectrum flatness limit for the right 80 MHz segment (segment 2) . This index is only relevant for bandwidths 80 MHz. Value 7 to 10: subcarrier indices of the trace margins to the lower spectrum flatness limit for the right 80 MHz segment (segment 2) . Values 6 to 10 are only relevant for bandwidths 80 MHz.

read(*mimo=Mimo.Default*) → List[int]

```
# SCPI: READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:MIMO<n>:X:AVERage
value: List[int] = driver.wlanMeas.multiEval.spectrFlatness.mimo.x.average.
↪read(mimo = repcap.Mimo.Default)
```

Return the subcarrier indices (x positions of the worst values) for the current, average, minimum and maximum margin values for true MIMO, antenna/stream number <n>. For the queries of spectrum flatness margins, see: method RsCMPX_WlanMeas.WlanMeas.MultiEval.SpectrFlatness.Mimo.Current.fetch etc.

Suppressed linked return values: reliability

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
spec_flat_margins_segments_tx: Up to 10 comma-separated values of subcarrier indices (one index per subcarrier range from left to right) Value 1: subcarrier index of the trace margin to the upper spectrum flatness limit For bandwidths 80 MHz, the value is only relevant for the left 80 MHz segment (segment 1) . Value 2 to 5: subcarrier indices, for the trace margin to the lower spectrum flatness limit For bandwidths 80 MHz, the values are only relevant for the left 80 MHz segment (segment 1) . Value 6: subcarrier index of the trace margin to the upper spectrum flatness limit for the right 80 MHz segment (segment 2) . This index is only relevant for bandwidths 80 MHz. Value 7 to 10: subcarrier indices of the trace margins to the lower spectrum flatness limit for the right 80 MHz segment (segment 2) . Values 6 to 10 are only relevant for bandwidths 80 MHz.

6.4.1.7.4.7 Current

SCPI Commands :

```
READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:MIMO<n>:X:CURRent
FETCh:WLAN:MEASurement<instance>:MEValuation:SFLatness:MIMO<n>:X:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*mimo=Mimo.Default*) → List[int]

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SFLatness:MIMO<n>:X:CURRent
value: List[int] = driver.wlanMeas.multiEval.spectrFlatness.mimo.x.current.
↪fetch(mimo = repcap.Mimo.Default)
```

Return the subcarrier indices (x positions of the worst values) for the current, average, minimum and maximum margin values for true MIMO, antenna/stream number <n>. For the queries of spectrum flatness margins, see: method RsCMPX_WlanMeas.WlanMeas.MultiEval.SpectrFlatness.Mimo.Current.fetch etc.

Suppressed linked return values: reliability

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
spec_flat_margins_segments_tx: Up to 10 comma-separated values of subcarrier indices (one index per subcarrier range from left to right) Value 1: subcarrier index of the trace margin to the upper spectrum flatness limit For bandwidths 80 MHz, the value is only relevant for the left 80 MHz segment (segment 1) . Value 2 to 5: subcarrier indices, for the trace margin to the lower spectrum flatness limit For bandwidths 80 MHz, the values are only relevant for the left 80 MHz segment (segment 1) . Value 6: subcarrier index of the trace margin to the upper spectrum flatness limit for the right 80 MHz segment (segment 2) . This index is only relevant for bandwidths 80 MHz. Value 7 to 10: subcarrier indices of the trace margins to the lower spectrum flatness limit for the right 80 MHz segment (segment 2) . Values 6 to 10 are only relevant for bandwidths 80 MHz.

read(*mimo=Mimo.Default*) → List[int]

```
# SCPI: READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:MIMO<n>:X:CURRent
value: List[int] = driver.wlanMeas.multiEval.spectrFlatness.mimo.x.current.
↪read(mimo = repcap.Mimo.Default)
```

Return the subcarrier indices (x positions of the worst values) for the current, average, minimum and maximum margin values for true MIMO, antenna/stream number <n>. For the queries of spectrum flatness margins, see: method RsCMPX_WlanMeas.WlanMeas.MultiEval.SpectrFlatness.Mimo.Current.fetch etc.

Suppressed linked return values: reliability

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
spec_flat_margins_segments_tx: Up to 10 comma-separated values of subcarrier indices (one index per subcarrier range from left to right) Value 1: subcarrier index of the trace margin to the upper spectrum flatness limit For bandwidths 80 MHz, the value is only relevant for the left 80 MHz segment (segment 1) . Value 2 to 5: subcarrier indices, for the trace margin to the lower spectrum flatness limit For bandwidths 80 MHz, the values are only relevant for the left 80 MHz segment (segment 1) . Value 6: subcarrier index of the trace margin to the upper spectrum flatness limit for the right 80 MHz segment (segment 2) . This index is only relevant for bandwidths 80 MHz. Value 7 to 10: subcarrier indices of the trace margins to the lower spectrum flatness limit for the right 80 MHz segment (segment 2) . Values 6 to 10 are only relevant for bandwidths 80 MHz.

6.4.1.7.4.8 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:MIMO<n>:X:MAXimum
FETCh:WLAN:MEASurement<instance>:MEValuation:SFLatness:MIMO<n>:X:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*mimo=Mimo.Default*) → List[int]

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SFLatness:MIMO<n>:X:MAXimum
value: List[int] = driver.wlanMeas.multiEval.spectrFlatness.mimo.x.maximum.
↪fetch(mimo = repcap.Mimo.Default)
```

Return the subcarrier indices (x positions of the worst values) for the current, average, minimum and maximum margin values for true MIMO, antenna/stream number <n>. For the queries of spectrum flatness margins, see: method RsCMPX_WlanMeas.WlanMeas.MultiEval.SpectrFlatness.Mimo.Current.fetch etc.

Suppressed linked return values: reliability

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
spec_flat_margins_segments_tx: Up to 10 comma-separated values of subcarrier indices (one index per subcarrier range from left to right) Value 1: subcarrier index of the trace margin to the upper spectrum flatness limit For bandwidths 80 MHz, the value is only relevant for the left 80 MHz segment (segment 1) . Value 2 to 5: subcarrier indices, for the trace margin to the lower spectrum flatness limit For bandwidths 80 MHz, the values are only relevant for the left 80 MHz segment (segment 1) . Value 6: subcarrier index of the trace margin to the upper spectrum flatness limit for the right 80 MHz segment (segment 2) . This index is only relevant for bandwidths 80 MHz. Value 7 to 10: subcarrier indices of the trace margins to the lower spectrum flatness limit for the right 80 MHz segment (segment 2) . Values 6 to 10 are only relevant for bandwidths 80 MHz.

read(*mimo=Mimo.Default*) → List[int]

```
# SCPI: READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:MIMO<n>:X:MAXimum
value: List[int] = driver.wlanMeas.multiEval.spectrFlatness.mimo.x.maximum.
↪read(mimo = repcap.Mimo.Default)
```

Return the subcarrier indices (x positions of the worst values) for the current, average, minimum and maximum margin values for true MIMO, antenna/stream number <n>. For the queries of spectrum flatness margins, see: method RsCMPX_WlanMeas.WlanMeas.MultiEval.SpectrFlatness.Mimo.Current.fetch etc.

Suppressed linked return values: reliability

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
spec_flat_margins_segments_tx: Up to 10 comma-separated values of subcarrier indices (one index per subcarrier range from left to right) Value 1: subcarrier index of the trace margin to the upper spectrum flatness limit For bandwidths 80 MHz, the value is only relevant for the left 80 MHz segment (segment 1) . Value 2 to 5: subcarrier indices, for the trace margin to the lower spectrum flatness limit For bandwidths 80 MHz, the values are only relevant for the left 80 MHz segment (segment 1) . Value 6: subcarrier index of the trace margin to the upper spectrum flatness limit for the right 80 MHz segment (segment 2) . This index is only relevant for bandwidths 80 MHz. Value 7 to 10: subcarrier indices of the trace margins to the lower spectrum flatness limit for the right 80 MHz segment (segment 2) . Values 6 to 10 are only relevant for bandwidths 80 MHz.

6.4.1.7.4.9 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:MIMO<n>:X:MINimum
FETCh:WLAN:MEASurement<instance>:MEValuation:SFLatness:MIMO<n>:X:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*mimo=Mimo.Default*) → List[int]

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SFLatness:MIMO<n>:X:MINimum
value: List[int] = driver.wlanMeas.multiEval.spectrFlatness.mimo.x.minimum.
↪fetch(mimo = repcap.Mimo.Default)
```

Return the subcarrier indices (x positions of the worst values) for the current, average, minimum and maximum margin values for true MIMO, antenna/stream number <n>. For the queries of spectrum flatness margins, see: method RsCMPX_WlanMeas.WlanMeas.MultiEval.SpectrFlatness.Mimo.Current.fetch etc.

Suppressed linked return values: reliability

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
spec_flat_margins_segments_tx: Up to 10 comma-separated values of subcarrier indices (one index per subcarrier range from left to right) Value 1: subcarrier index of the trace margin to the upper spectrum flatness limit For bandwidths 80 MHz, the value is only relevant for the left 80 MHz segment (segment 1) . Value 2 to 5: subcarrier indices, for the trace margin to the lower spectrum flatness limit For bandwidths 80 MHz, the values are only relevant for the left 80 MHz segment (segment 1) . Value 6: subcarrier index of the trace margin to the upper spectrum flatness limit for the right 80 MHz segment (segment 2) . This index is only relevant for bandwidths 80 MHz. Value 7 to 10: subcarrier indices of the trace margins to the lower spectrum flatness limit for the right 80 MHz segment (segment 2) . Values 6 to 10 are only relevant for bandwidths 80 MHz.

read(*mimo=Mimo.Default*) → List[int]

```
# SCPI: READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:MIMO<n>:X:MINimum
value: List[int] = driver.wlanMeas.multiEval.spectrFlatness.mimo.x.minimum.
↪read(mimo = repcap.Mimo.Default)
```

Return the subcarrier indices (x positions of the worst values) for the current, average, minimum and maximum margin values for true MIMO, antenna/stream number <n>. For the queries of spectrum flatness margins, see: method RsCMPX_WlanMeas.WlanMeas.MultiEval.SpectrFlatness.Mimo.Current.fetch etc.

Suppressed linked return values: reliability

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
spec_flat_margins_segments_tx: Up to 10 comma-separated values of subcarrier in-

dices (one index per subcarrier range from left to right) Value 1: subcarrier index of the trace margin to the upper spectrum flatness limit For bandwidths 80 MHz, the value is only relevant for the left 80 MHz segment (segment 1) . Value 2 to 5: subcarrier indices, for the trace margin to the lower spectrum flatness limit For bandwidths 80 MHz, the values are only relevant for the left 80 MHz segment (segment 1) . Value 6: subcarrier index of the trace margin to the upper spectrum flatness limit for the right 80 MHz segment (segment 2) . This index is only relevant for bandwidths 80 MHz. Value 7 to 10: subcarrier indices of the trace margins to the lower spectrum flatness limit for the right 80 MHz segment (segment 2) . Values 6 to 10 are only relevant for bandwidths 80 MHz.

6.4.1.7.5 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:MINimum
FETCh:WLAN:MEASurement<instance>:MEValuation:SFLatness:MINimum
CALCulate:WLAN:MEASurement<instance>:MEValuation:SFLatness:MINimum
```

class MinimumCls

Minimum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate() → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<instance>:MEValuation:SFLatness:MINimum
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.spectrFlatness.
↪minimum.calculate()
```

Returns the margin values of the spectrum flatness measurement for the current, average, minimum and maximum traces. A positive margin indicates a violation of the spectrum flatness limit. The respective trace value is located above the upper or below the lower limit line. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

return

margins: Up to 10 comma-separated list of margins (one value per subcarrier range from left to right) Value 1: trace margin to the upper spectrum flatness limit For bandwidths 80 MHz, the margin is only relevant for the left 80 MHz segment (segment 1) . Value 2 to 5: trace margins to the lower spectrum flatness limit For bandwidths 80 MHz, the margins are only relevant for the left 80 MHz segment (segment 1) . Value 6: trace margin to the upper spectrum flatness limit for the right 80 MHz segment (segment 2) . This margin is only relevant for bandwidths 80 MHz. Value 7 to 10: trace margins to the lower spectrum flatness limit for the right 80 MHz segment (segment 2) . Values 6 to 10 are only relevant for bandwidths 80 MHz.

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SFLatness:MINimum
value: List[float] = driver.wlanMeas.multiEval.spectrFlatness.minimum.fetch()
```

Returns the margin values of the spectrum flatness measurement for the current, average, minimum and maximum traces. A positive margin indicates a violation of the spectrum flatness limit. The respective

trace value is located above the upper or below the lower limit line. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

return
margins: Up to 10 comma-separated list of margins (one value per subcarrier range from left to right) Value 1: trace margin to the upper spectrum flatness limit For bandwidths 80 MHz, the margin is only relevant for the left 80 MHz segment (segment 1) . Value 2 to 5: trace margins to the lower spectrum flatness limit For bandwidths 80 MHz, the margins are only relevant for the left 80 MHz segment (segment 1) . Value 6: trace margin to the upper spectrum flatness limit for the right 80 MHz segment (segment 2) . This margin is only relevant for bandwidths 80 MHz. Value 7 to 10: trace margins to the lower spectrum flatness limit for the right 80 MHz segment (segment 2) . Values 6 to 10 are only relevant for bandwidths 80 MHz.

read() → List[float]

```
# SCPI: READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:MINimum
value: List[float] = driver.wlanMeas.multiEval.spectrFlatness.minimum.read()
```

Returns the margin values of the spectrum flatness measurement for the current, average, minimum and maximum traces. A positive margin indicates a violation of the spectrum flatness limit. The respective trace value is located above the upper or below the lower limit line. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

return
margins: Up to 10 comma-separated list of margins (one value per subcarrier range from left to right) Value 1: trace margin to the upper spectrum flatness limit For bandwidths 80 MHz, the margin is only relevant for the left 80 MHz segment (segment 1) . Value 2 to 5: trace margins to the lower spectrum flatness limit For bandwidths 80 MHz, the margins are only relevant for the left 80 MHz segment (segment 1) . Value 6: trace margin to the upper spectrum flatness limit for the right 80 MHz segment (segment 2) . This margin is only relevant for bandwidths 80 MHz. Value 7 to 10: trace margins to the lower spectrum flatness limit for the right 80 MHz segment (segment 2) . Values 6 to 10 are only relevant for bandwidths 80 MHz.

6.4.1.7.6 X

class XCls

X commands group definition. 8 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.spectrFlatness.x.clone()
```

Subgroups

6.4.1.7.6.1 Average

SCPI Commands :

```
READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:X:AVERage
FETCh:WLAN:MEASurement<instance>:MEValuation:SFLatness:X:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[int]

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SFLatness:X:AVERage
value: List[int] = driver.wlanMeas.multiEval.spectrFlatness.x.average.fetch()
```

Return the subcarrier indices (x positions of the worst values) for the current, average, minimum and maximum margin values.

Suppressed linked return values: reliability

return

margins: Up to 10 comma-separated values of subcarrier indices (one index per subcarrier range from left to right) Value 1: subcarrier index of the trace margin to the upper spectrum flatness limit For bandwidths 80 MHz, the value is only relevant for the left 80 MHz segment (segment 1) . Value 2 to 5: subcarrier indices, for the trace margin to the lower spectrum flatness limit For bandwidths 80 MHz, the values are only relevant for the left 80 MHz segment (segment 1) . Value 6: subcarrier index of the trace margin to the upper spectrum flatness limit for the right 80 MHz segment (segment 2) . This index is only relevant for bandwidths 80 MHz. Value 7 to 10: subcarrier indices of the trace margins to the lower spectrum flatness limit for the right 80 MHz segment (segment 2) . Values 6 to 10 are only relevant for bandwidths 80 MHz.

read() → List[int]

```
# SCPI: READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:X:AVERage
value: List[int] = driver.wlanMeas.multiEval.spectrFlatness.x.average.read()
```

Return the subcarrier indices (x positions of the worst values) for the current, average, minimum and maximum margin values.

Suppressed linked return values: reliability

return
margins: Up to 10 comma-separated values of subcarrier indices (one index per subcarrier range from left to right) Value 1: subcarrier index of the trace margin to the upper spectrum flatness limit For bandwidths 80 MHz, the value is only relevant for the left 80 MHz segment (segment 1) . Value 2 to 5: subcarrier indices, for the trace margin to the lower spectrum flatness limit For bandwidths 80 MHz, the values are only relevant for the left 80 MHz segment (segment 1) . Value 6: subcarrier index of the trace margin to the upper spectrum flatness limit for the right 80 MHz segment (segment 2) . This index is only relevant for bandwidths 80 MHz. Value 7 to 10: subcarrier indices of the trace margins to the lower spectrum flatness limit for the right 80 MHz segment (segment 2) . Values 6 to 10 are only relevant for bandwidths 80 MHz.

6.4.1.7.6.2 Current

SCPI Commands :

```
READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:X:CURRent
FETCh:WLAN:MEASurement<instance>:MEValuation:SFLatness:X:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[int]

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SFLatness:X:CURRent
value: List[int] = driver.wlanMeas.multiEval.spectrFlatness.x.current.fetch()
```

Return the subcarrier indices (x positions of the worst values) for the current, average, minimum and maximum margin values.

Suppressed linked return values: reliability

return
margins: Up to 10 comma-separated values of subcarrier indices (one index per subcarrier range from left to right) Value 1: subcarrier index of the trace margin to the upper spectrum flatness limit For bandwidths 80 MHz, the value is only relevant for the left 80 MHz segment (segment 1) . Value 2 to 5: subcarrier indices, for the trace margin to the lower spectrum flatness limit For bandwidths 80 MHz, the values are only relevant for the left 80 MHz segment (segment 1) . Value 6: subcarrier index of the trace margin to the upper spectrum flatness limit for the right 80 MHz segment (segment 2) . This index is only relevant for bandwidths 80 MHz. Value 7 to 10: subcarrier indices of the trace margins to the lower spectrum flatness limit for the right 80 MHz segment (segment 2) . Values 6 to 10 are only relevant for bandwidths 80 MHz.

read() → List[int]

```
# SCPI: READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:X:CURRent
value: List[int] = driver.wlanMeas.multiEval.spectrFlatness.x.current.read()
```

Return the subcarrier indices (x positions of the worst values) for the current, average, minimum and maximum margin values.

Suppressed linked return values: reliability

return
margins: Up to 10 comma-separated values of subcarrier indices (one index per subcarrier range from left to right) Value 1: subcarrier index of the trace margin to the upper spectrum flatness limit For bandwidths 80 MHz, the value is only relevant for the left 80 MHz segment (segment 1) . Value 2 to 5: subcarrier indices, for the trace margin to the lower spectrum flatness limit For bandwidths 80 MHz, the values are only relevant for the left 80 MHz segment (segment 1) . Value 6: subcarrier index of the trace margin to the upper spectrum flatness limit for the right 80 MHz segment (segment 2) . This index is only relevant for bandwidths 80 MHz. Value 7 to 10: subcarrier indices of the trace margins to the lower spectrum flatness limit for the right 80 MHz segment (segment 2) . Values 6 to 10 are only relevant for bandwidths 80 MHz.

6.4.1.7.6.3 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:X:MAXimum
FETCh:WLAN:MEASurement<instance>:MEValuation:SFLatness:X:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[int]

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SFLatness:X:MAXimum
value: List[int] = driver.wlanMeas.multiEval.spectrFlatness.x.maximum.fetch()
```

Return the subcarrier indices (x positions of the worst values) for the current, average, minimum and maximum margin values.

Suppressed linked return values: reliability

return
margins: Up to 10 comma-separated values of subcarrier indices (one index per subcarrier range from left to right) Value 1: subcarrier index of the trace margin to the upper spectrum flatness limit For bandwidths 80 MHz, the value is only relevant for the left 80 MHz segment (segment 1) . Value 2 to 5: subcarrier indices, for the trace margin to the lower spectrum flatness limit For bandwidths 80 MHz, the values are only relevant for the left 80 MHz segment (segment 1) . Value 6: subcarrier index of the trace margin to the upper spectrum flatness limit for the right 80 MHz segment (segment 2) . This index is only relevant for bandwidths 80 MHz. Value 7 to 10: subcarrier indices of the trace margins to the lower spectrum flatness limit for the right 80 MHz segment (segment 2) . Values 6 to 10 are only relevant for bandwidths 80 MHz.

read() → List[int]

```
# SCPI: READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:X:MAXimum
value: List[int] = driver.wlanMeas.multiEval.spectrFlatness.x.maximum.read()
```

Return the subcarrier indices (x positions of the worst values) for the current, average, minimum and maximum margin values.

Suppressed linked return values: reliability

return
margins: Up to 10 comma-separated values of subcarrier indices (one index per subcarrier range from left to right) Value 1: subcarrier index of the trace margin to the upper spectrum flatness limit For bandwidths 80 MHz, the value is only relevant for the left 80 MHz segment (segment 1) . Value 2 to 5: subcarrier indices, for the trace margin to the lower spectrum flatness limit For bandwidths 80 MHz, the values are only relevant for the left 80 MHz segment (segment 1) . Value 6: subcarrier index of the trace margin to the upper spectrum flatness limit for the right 80 MHz segment (segment 2) . This index is only relevant for bandwidths 80 MHz. Value 7 to 10: subcarrier indices of the trace margins to the lower spectrum flatness limit for the right 80 MHz segment (segment 2) . Values 6 to 10 are only relevant for bandwidths 80 MHz.

6.4.1.7.6.4 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:X:MINimum
FETCh:WLAN:MEASurement<instance>:MEValuation:SFLatness:X:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[int]

```
# SCPI: FETCh:WLAN:MEASurement<instance>:MEValuation:SFLatness:X:MINimum
value: List[int] = driver.wlanMeas.multiEval.spectrFlatness.x.minimum.fetch()
```

Return the subcarrier indices (x positions of the worst values) for the current, average, minimum and maximum margin values.

Suppressed linked return values: reliability

return
margins: Up to 10 comma-separated values of subcarrier indices (one index per subcarrier range from left to right) Value 1: subcarrier index of the trace margin to the upper spectrum flatness limit For bandwidths 80 MHz, the value is only relevant for the left 80 MHz segment (segment 1) . Value 2 to 5: subcarrier indices, for the trace margin to the lower spectrum flatness limit For bandwidths 80 MHz, the values are only relevant for the left 80 MHz segment (segment 1) . Value 6: subcarrier index of the trace margin to the upper spectrum flatness limit for the right 80 MHz segment (segment 2) . This index is only relevant for bandwidths 80 MHz. Value 7 to 10: subcarrier indices of the trace margins to the lower spectrum flatness limit for the right 80 MHz segment (segment 2) . Values 6 to 10 are only relevant for bandwidths 80 MHz.

read() → List[int]

```
# SCPI: READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:X:MINimum
value: List[int] = driver.wlanMeas.multiEval.spectrFlatness.x.minimum.read()
```

Return the subcarrier indices (x positions of the worst values) for the current, average, minimum and maximum margin values.

Suppressed linked return values: reliability

return
margins: Up to 10 comma-separated values of subcarrier indices (one index per subcarrier range from left to right) Value 1: subcarrier index of the trace margin to the upper spectrum flatness limit For bandwidths 80 MHz, the value is only relevant for the left 80 MHz segment (segment 1) . Value 2 to 5: subcarrier indices, for the trace margin to the lower spectrum flatness limit For bandwidths 80 MHz, the values are only relevant for the left 80 MHz segment (segment 1) . Value 6: subcarrier index of the trace margin to the upper spectrum flatness limit for the right 80 MHz segment (segment 2) . This index is only relevant for bandwidths 80 MHz. Value 7 to 10: subcarrier indices of the trace margins to the lower spectrum flatness limit for the right 80 MHz segment (segment 2) . Values 6 to 10 are only relevant for bandwidths 80 MHz.

6.4.1.8 State

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:STATe
```

class StateCls

State commands group definition. 2 total commands, 1 Subgroups, 1 group commands

fetch() → ResourceState

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:STATe
value: enums.ResourceState = driver.wlanMeas.multiEval.state.fetch()
```

Queries the main measurement state. Without query parameters, the state is returned immediately. With query parameters, the state is returned when the <TargetMainState> and the <TargetSyncState> are reached or when the <Timeout> expires.

return
multi_eval_state: Current state or target state of ongoing state transition OFF: measurement off RUN: measurement running RDY: measurement completed

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.state.clone()
```

Subgroups

6.4.1.8.1 All

SCPI Command :

```
FETCh:WLAN:MEASurement<Instance>:MEValuation:STATe:ALL
```

class AllCls

All commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Main_State: enums.ResourceState: Current state or target state of ongoing state transition OFF: measurement off RUN: measurement running RDY: measurement completed
- Sync_State: enums.ResourceState: PEND: transition to MainState ongoing ADJ: MainState reached
- Res_State: enums.ResourceState: QUE: waiting for resource allocation ACT: resources allocated INV: no resources allocated

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:STATe:ALL
value: FetchStruct = driver.wlanMeas.multiEval.state.all.fetch()
```

Queries the main measurement state and the measurement substates. Without query parameters, the states are returned immediately. With query parameters, the states are returned when the <TargetMainState> and the <TargetSyncState> are reached or when the <Timeout> expires.

return
structure: for return value, see the help for FetchStruct structure arguments.

6.4.1.9 Trace

class TraceCls

Trace commands group definition. 360 total commands, 7 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.clone()
```

Subgroups

6.4.1.9.1 CfError

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:CFERror
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:CFERror
```

class CfErrorCls

CfError commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:CFERror
value: List[float] = driver.wlanMeas.multiEval.trace.cfError.fetch(start = 1.0,␣
↪count = 1.0, decimation = 1.0)
```

Return carrier frequency offset (CFO) error traces for HE and EHT. The number of results corresponds to the statistic count, see method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.Scount.modulation.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
cfo_frequencies: Comma-separated list of CFO error values

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:CFERror
value: List[float] = driver.wlanMeas.multiEval.trace.cfError.read(start = 1.0,␣
↪count = 1.0, decimation = 1.0)
```

Return carrier frequency offset (CFO) error traces for HE and EHT. The number of results corresponds to the statistic count, see method RsCMPX_WlanMeas.Configure.WlanMeas.MultiEval.Scount.modulation.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
cfo_frequencies: Comma-separated list of CFO error values

6.4.1.9.2 EvMagnitude

class EvMagnitudeCls

EvMagnitude commands group definition. 84 total commands, 6 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.evMagnitude.clone()
```

Subgroups

6.4.1.9.2.1 Acsiso

class AcsisoCls

Acsiso commands group definition. 6 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.evMagnitude.acsiso.clone()
```

Subgroups

6.4.1.9.2.2 Symbol

class SymbolCls

Symbol commands group definition. 6 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.evMagnitude.acsiso.symbol.clone()
```

Subgroups

6.4.1.9.2.3 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:ACSiso:SYMBol:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:ACSiso:SYMBol:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:ACSiso:SYMBol:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.acsiso.symbol.
↪average.fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

return
evm_aver: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:ACSiso:SYMBol:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.acsiso.symbol.
↪average.read(start = 1.0, count = 1.0, decimation = 1.0)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

return
evm_aver: No help available

6.4.1.9.2.4 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:ACSiso:SYMBol:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:ACSiso:SYMBol:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:ACSiso:SYMBol:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.acsiso.symbol.
↪current.fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

return
evm_curr: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:ACSiso:SYMBol:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.acsiso.symbol.
↪current.read(start = 1.0, count = 1.0, decimation = 1.0)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

return
evm_curr: No help available

6.4.1.9.2.5 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:ACSiso:SYMBol:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:ACSiso:SYMBol:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:ACSiso:SYMBol:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.acsiso.symbol.
↪maximum.fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

return
evm_max: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:ACSiso:SYMBol:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.acsiso.symbol.
↪maximum.read(start = 1.0, count = 1.0, decimation = 1.0)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

return
evm_max: No help available

6.4.1.9.2.6 Carrier

class CarrierCls

Carrier commands group definition. 32 total commands, 6 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.evMagnitude.carrier.clone()
```

Subgroups

6.4.1.9.2.7 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:CARRier:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:CARRier:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:CARRier:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.carrier.
↪average.fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the EVM vs subcarrier traces for SISO connections. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
evm_trace_aver: Comma-separated list of 2n+1 values, for subcarrier -n to subcarrier +n (including data, pilot and unused subcarriers / for 802.11ax, excluding subcarrier -3 to subcarrier 3) n depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:CARRier:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.carrier.
↪average.read(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the EVM vs subcarrier traces for SISO connections. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
evm_trace_aver: Comma-separated list of 2n+1 values, for subcarrier -n to subcarrier +n (including data, pilot and unused subcarriers / for 802.11ax, excluding subcarrier -3 to subcarrier 3) n depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

6.4.1.9.2.8 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:CARRier:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:CARRier:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:CARRier:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.carrier.
↪current.fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the EVM vs subcarrier traces for SISO connections. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
evm_trace_curr: Comma-separated list of 2n+1 values, for subcarrier -n to subcarrier +n (including data, pilot and unused subcarriers / for 802.11ax, excluding subcarrier -3 to subcarrier 3) n depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:CARRier:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.carrier.
↪current.read(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the EVM vs subcarrier traces for SISO connections. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
evm_trace_curr: Comma-separated list of 2n+1 values, for subcarrier -n to subcarrier +n (including data, pilot and unused subcarriers / for 802.11ax, excluding subcarrier -3 to subcarrier 3) n depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

6.4.1.9.2.9 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:CARRier:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:CARRier:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:CARRier:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.carrier.
↪maximum.fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the EVM vs subcarrier traces for SISO connections. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
evm_trace_max: Comma-separated list of 2n+1 values, for subcarrier -n to subcarrier +n (including data, pilot and unused subcarriers / for 802.11ax, excluding subcarrier -3 to subcarrier 3) n depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:CARRier:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.carrier.
↪maximum.read(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the EVM vs subcarrier traces for SISO connections. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
evm_trace_max: Comma-separated list of 2n+1 values, for subcarrier -n to subcarrier +n (including data, pilot and unused subcarriers / for 802.11ax, excluding subcarrier -3 to subcarrier 3) n depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

6.4.1.9.2.10 Mimo<Mimo>

RepCap Settings

```
# Range: Nr1 .. Nr8
rc = driver.wlanMeas.multiEval.trace.evMagnitude.carrier.mimo.repcap_mimo_get()
driver.wlanMeas.multiEval.trace.evMagnitude.carrier.mimo.repcap_mimo_set(repcap.Mimo.Nr1)
```

class MimoCls

Mimo commands group definition. 16 total commands, 5 Subgroups, 0 group commands Repeated Capability: Mimo, default value after init: Mimo.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.evMagnitude.carrier.mimo.clone()
```

Subgroups

6.4.1.9.2.11 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:CARRier:MIMO<n>:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:CARRier:MIMO<n>:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:CARRier:MIMO<n>:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.carrier.mimo.
↪average.fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.
↪Default)
```

Return the values of the EVM vs subcarrier traces for MIMO connections. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
evm_vs_carr_avg: Comma-separated list of 2n+1 values, for subcarrier -n to subcarrier +n (including data, pilot and unused subcarriers / for 802.11ax, excluding subcarrier -3 to subcarrier 3) n depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:CARRier:MIMO<n>:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.carrier.mimo.
↪average.read(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.
↪Default)
```

Return the values of the EVM vs subcarrier traces for MIMO connections. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
evm_vs_carr_avg: Comma-separated list of 2n+1 values, for subcarrier -n to subcarrier +n (including data, pilot and unused subcarriers / for 802.11ax, excluding subcarrier -3 to subcarrier 3) n depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

6.4.1.9.2.12 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:CARRier:MIMO<n>:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:CARRier:MIMO<n>:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:CARRier:MIMO<n>:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.carrier.mimo.
↪current.fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.
↪Default)
```

Return the values of the EVM vs subcarrier traces for MIMO connections. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
evm_vs_carr_cur: Comma-separated list of 2n+1 values, for subcarrier -n to subcarrier +n (including data, pilot and unused subcarriers / for 802.11ax, excluding subcarrier -3 to subcarrier 3) n depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:CARRier:MIMO<n>:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.carrier.mimo.
↪current.read(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.
↪Default)
```

Return the values of the EVM vs subcarrier traces for MIMO connections. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
evm_vs_carr_cur: Comma-separated list of 2n+1 values, for subcarrier -n to subcarrier +n (including data, pilot and unused subcarriers / for 802.11ax, excluding subcarrier -3 to subcarrier 3) n depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

6.4.1.9.2.13 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:CARRier:MIMO<n>:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:CARRier:MIMO<n>:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:CARRier:MIMO<n>:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.carrier.mimo.
↪maximum.fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.
↪Default)
```

Return the values of the EVM vs subcarrier traces for MIMO connections. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
evm_vs_carr_max: Comma-separated list of 2n+1 values, for subcarrier -n to subcarrier +n (including data, pilot and unused subcarriers / for 802.11ax, excluding subcarrier -3 to subcarrier 3) n depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:CARRier:MIMO<n>:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.carrier.mimo.
↪maximum.read(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.
↪Default)
```

Return the values of the EVM vs subcarrier traces for MIMO connections. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
evm_vs_carr_max: Comma-separated list of 2n+1 values, for subcarrier -n to subcarrier +n (including data, pilot and unused subcarriers / for 802.11ax, excluding subcarrier -3 to subcarrier 3) n depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

6.4.1.9.2.14 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:CARRier:MIMO<n>:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:CARRier:MIMO<n>:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:CARRier:MIMO<n>:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.carrier.mimo.
↪minimum.fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.
↪Default)
```

Return the values of the EVM vs subcarrier traces for MIMO connections. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
evm_vs_carr_min: Comma-separated list of 2n+1 values, for subcarrier -n to subcarrier +n (including data, pilot and unused subcarriers / for 802.11ax, excluding subcarrier -3 to subcarrier 3) n depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:CARRier:MIMO<n>:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.carrier.mimo.
↪minimum.read(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.
↪Default)
```

Return the values of the EVM vs subcarrier traces for MIMO connections. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
evm_vs_carr_min: Comma-separated list of 2n+1 values, for subcarrier -n to subcarrier +n (including data, pilot and unused subcarriers / for 802.11ax, excluding subcarrier -3 to subcarrier 3) n depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

6.4.1.9.2.15 Segment

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.wlanMeas.multiEval.trace.evMagnitude.carrier.mimo.segment.repcap_segment_
↪get()
driver.wlanMeas.multiEval.trace.evMagnitude.carrier.mimo.segment.repcap_segment_
↪set(repcap.Segment.Nr1)
```

class SegmentCls

Segment commands group definition. 8 total commands, 4 Subgroups, 0 group commands Repeated Capability: Segment, default value after init: Segment.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.evMagnitude.carrier.mimo.segment.clone()
```

Subgroups

6.4.1.9.2.16 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:CARRier:MIMO<n>:SEGMent
↪<seg>:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:CARRier:MIMO<n>:SEGMent
↪<seg>:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:CARRier:MIMO<n>:SEGMent<seg>:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.carrier.mimo.
↪segment.average.fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo =␣
↪repcap.Mimo.Default, segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
evm_vs_carr_avg: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:CARRier:MIMO<n>:SEGMent<seg>:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.carrier.mimo.
↪segment.average.read(start = 1.0, count = 1.0, decimation = 1.0, mimo =␣
↪repcap.Mimo.Default, segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
evm_vs_carr_avg: No help available

6.4.1.9.2.17 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:CARRier:MIMO<n>:SEGMent
↪<seg>:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:CARRier:MIMO<n>:SEGMent
↪<seg>:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:CARRier:MIMO<n>:SEGMent<seg>:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.carrier.mimo.
↪segment.current.fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo =␣
↪repcap.Mimo.Default, segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
evm_vs_carr_cur: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:CARRier:MIMO<n>:SEGMent<seg>:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.carrier.mimo.
↪segment.current.read(start = 1.0, count = 1.0, decimation = 1.0, mimo =␣
↪repcap.Mimo.Default, segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
evm_vs_carr_cur: No help available

6.4.1.9.2.18 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:CARRier:MIMO<n>:SEGMent
↪<seg>:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:CARRier:MIMO<n>:SEGMent
↪<seg>:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:CARRier:MIMO<n>:SEGMent<seg>:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.carrier.mimo.
↪segment.maximum.fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo =␣
↪repcap.Mimo.Default, segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
evm_vs_carr_max: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:CARRier:MIMO<n>:SEGMent<seg>:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.carrier.mimo.
↪segment.maximum.read(start = 1.0, count = 1.0, decimation = 1.0, mimo =␣
↪repcap.Mimo.Default, segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
evm_vs_carr_max: No help available

6.4.1.9.2.19 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:CARRier:MIMO<n>:SEGMent
↪<seg>:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:CARRier:MIMO<n>:SEGMent
↪<seg>:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:CARRier:MIMO<n>:SEGMent<seg>:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.carrier.mimo.
↪segment.minimum.fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo =␣
↪repcap.Mimo.Default, segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
evm_vs_carr_min: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:CARRier:MIMO<n>:SEGMent<seg>:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.carrier.mimo.
↪segment.minimum.read(start = 1.0, count = 1.0, decimation = 1.0, mimo =␣
↪repcap.Mimo.Default, segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
evm_vs_carr_min: No help available

6.4.1.9.2.20 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:CARRier:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:CARRier:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:CARRier:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.carrier.
↪minimum.fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the EVM vs subcarrier traces for SISO connections. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
evm_trace_min: Comma-separated list of 2n+1 values, for subcarrier -n to subcarrier +n (including data, pilot and unused subcarriers / for 802.11ax, excluding subcarrier -3 to subcarrier 3) n depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:CARRier:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.carrier.
↪minimum.read(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the EVM vs subcarrier traces for SISO connections. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
evm_trace_min: Comma-separated list of 2n+1 values, for subcarrier -n to subcarrier +n (including data, pilot and unused subcarriers / for 802.11ax, excluding subcarrier -3 to subcarrier 3) n depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

6.4.1.9.2.21 Segment

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.wlanMeas.multiEval.trace.evMagnitude.carrier.segment.repcap_segment_get()
driver.wlanMeas.multiEval.trace.evMagnitude.carrier.segment.repcap_segment_set(repcap.
↪Segment.Nr1)
```

class SegmentCls

Segment commands group definition. 8 total commands, 4 Subgroups, 0 group commands Repeated Capability: Segment, default value after init: Segment.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.evMagnitude.carrier.segment.clone()
```

Subgroups

6.4.1.9.2.22 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:CARRier:SEGMent<seg>
↪:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:CARRier:SEGMent<seg>
↪:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:CARRier:SEGMent<seg>:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.carrier.
↪segment.average.fetch(start = 1.0, count = 1.0, decimation = 1.0, segment =␣
↪repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
evm_vs_carr_avg: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:CARRier:SEGMent<seg>:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.carrier.
↪segment.average.read(start = 1.0, count = 1.0, decimation = 1.0, segment =␣
↪repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
evm_vs_carr_avg: No help available

6.4.1.9.2.23 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:CARRier:SEGMent<seg>
↪:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:CARRier:SEGMent<seg>
↪:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:CARRier:SEGMent<seg>:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.carrier.
↪segment.current.fetch(start = 1.0, count = 1.0, decimation = 1.0, segment =␣
↪repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
evm_vs_carr_cur: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:CARRier:SEGMent<seg>:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.carrier.
↪segment.current.read(start = 1.0, count = 1.0, decimation = 1.0, segment =␣
↪repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
evm_vs_carr_cur: No help available

6.4.1.9.2.24 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:CARRier:SEGMent<seg>
↪:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:CARRier:SEGMent<seg>
↪:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:CARRier:SEGMent<seg>:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.carrier.
↪segment.maximum.fetch(start = 1.0, count = 1.0, decimation = 1.0, segment =␣
↪repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
evm_vs_carr_max: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:CARRier:SEGMent<seg>:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.carrier.
↪segment.maximum.read(start = 1.0, count = 1.0, decimation = 1.0, segment =␣
↪repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
evm_vs_carr_max: No help available

6.4.1.9.2.25 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:CARRier:SEGMent<seg>
↪:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:CARRier:SEGMent<seg>
↪:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:CARRier:SEGMent<seg>:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.carrier.
↪segment.minimum.fetch(start = 1.0, count = 1.0, decimation = 1.0, segment =␣
↪repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
evm_vs_carr_min: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:CARRier:SEGMent<seg>:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.carrier.
↪segment.minimum.read(start = 1.0, count = 1.0, decimation = 1.0, segment =␣
↪repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
evm_vs_carr_min: No help available

6.4.1.9.2.26 Dsss

class DsssCls

Dsss commands group definition. 6 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.evMagnitude.dsss.clone()
```

Subgroups

6.4.1.9.2.27 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:DSSS:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:DSSS:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:DSSS:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.dsss.average.
↪fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the EVM vs chip traces. The results of the current, average and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
evm_aver: Comma-separated list of EVM values, each value related to one chip. The maximum number of values is 1000.

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:DSSS:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.dsss.average.
↪read(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the EVM vs chip traces. The results of the current, average and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
evm_aver: Comma-separated list of EVM values, each value related to one chip. The maximum number of values is 1000.

6.4.1.9.2.28 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:DSSS:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:DSSS:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:DSSS:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.dsss.current.
↪fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the EVM vs chip traces. The results of the current, average and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
evm_curr: Comma-separated list of EVM values, each value related to one chip. The maximum number of values is 1000.

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:DSSS:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.dsss.current.
↪read(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the EVM vs chip traces. The results of the current, average and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
evm_curr: Comma-separated list of EVM values, each value related to one chip. The maximum number of values is 1000.

6.4.1.9.2.29 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:DSSS:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:DSSS:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:DSSS:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.dsss.maximum.
↪fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the EVM vs chip traces. The results of the current, average and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
evm_max: Comma-separated list of EVM values, each value related to one chip. The maximum number of values is 1000.

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:DSSS:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.dsss.maximum.
↪read(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the EVM vs chip traces. The results of the current, average and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
evm_max: Comma-separated list of EVM values, each value related to one chip. The maximum number of values is 1000.

6.4.1.9.2.30 Nsiso

class NsisoCls

Nsiso commands group definition. 12 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.evMagnitude.nsiso.clone()
```

Subgroups

6.4.1.9.2.31 Carrier

class CarrierCls

Carrier commands group definition. 6 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.evMagnitude.nsiso.carrier.clone()
```

Subgroups

6.4.1.9.2.32 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:NSISo:CARRier:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:NSISo:CARRier:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:NSISo:CARRier:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.nsiso.carrier.
↪average.fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

return
evm_aver: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:NSISo:CARRier:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.nsiso.carrier.
↪average.read(start = 1.0, count = 1.0, decimation = 1.0)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

return
evm_aver: No help available

6.4.1.9.2.33 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:NSISo:CARRier:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:NSISo:CARRier:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:NSISo:CARRier:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.nsiso.carrier.
↪current.fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

return
evm_curr: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:NSISo:CARRier:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.nsiso.carrier.
↪current.read(start = 1.0, count = 1.0, decimation = 1.0)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

return
evm_curr: No help available

6.4.1.9.2.34 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:NSISo:CARRier:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:NSISo:CARRier:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:NSISo:CARRier:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.nsiso.carrier.
↪maximum.fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

return
evm_max: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:NSISo:CARRier:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.nsiso.carrier.
↪maximum.read(start = 1.0, count = 1.0, decimation = 1.0)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

return
evm_max: No help available

6.4.1.9.2.35 Symbol

class SymbolCls

Symbol commands group definition. 6 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.evMagnitude.nsiso.symbol.clone()
```

Subgroups

6.4.1.9.2.36 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:NSISo:SYMBol:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:NSISo:SYMBol:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:NSISo:SYMBol:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.nsiso.symbol.
↪average.fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

return
evm_aver: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:NSISo:SYMBol:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.nsiso.symbol.
↪average.read(start = 1.0, count = 1.0, decimation = 1.0)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

return
evm_aver: No help available

6.4.1.9.2.37 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:NSISo:SYMBol:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:NSISo:SYMBol:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:NSISo:SYMBol:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.nsiso.symbol.
↪current.fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

return
evm_curr: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:NSISo:SYMBol:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.nsiso.symbol.
↪current.read(start = 1.0, count = 1.0, decimation = 1.0)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

return
evm_curr: No help available

6.4.1.9.2.38 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:NSISo:SYMBol:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:NSISo:SYMBol:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:NSISo:SYMBol:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.nsiso.symbol.
↪maximum.fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

return
evm_max: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:NSISo:SYMBol:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.nsiso.symbol.
↪maximum.read(start = 1.0, count = 1.0, decimation = 1.0)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

return
evm_max: No help available

6.4.1.9.2.39 Ofdm

class OfdmCls

Ofdm commands group definition. 12 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.evMagnitude.ofdm.clone()
```

Subgroups

6.4.1.9.2.40 Carrier

class CarrierCls

Carrier commands group definition. 6 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.evMagnitude.ofdm.carrier.clone()
```

Subgroups

6.4.1.9.2.41 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:OFDM:CARRier:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:OFDM:CARRier:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:OFDM:CARRier:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.ofdm.carrier.
↪average.fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

return
evm_aver: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:OFDM:CARRier:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.ofdm.carrier.
↪average.read(start = 1.0, count = 1.0, decimation = 1.0)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

return
evm_aver: No help available

6.4.1.9.2.42 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:OFDM:CARRier:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:OFDM:CARRier:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:OFDM:CARRier:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.ofdm.carrier.
↪current.fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

return
evm_curr: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:OFDM:CARRier:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.ofdm.carrier.
↪current.read(start = 1.0, count = 1.0, decimation = 1.0)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

return
evm_curr: No help available

6.4.1.9.2.43 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:OFDM:CARRier:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:OFDM:CARRier:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:OFDM:CARRier:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.ofdm.carrier.
↪maximum.fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

return
evm_max: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:OFDM:CARRier:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.ofdm.carrier.
↪maximum.read(start = 1.0, count = 1.0, decimation = 1.0)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

return
evm_max: No help available

6.4.1.9.2.44 Symbol

class SymbolCls

Symbol commands group definition. 6 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.evMagnitude.ofdm.symbol.clone()
```

Subgroups

6.4.1.9.2.45 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:OFDM:SYMBol:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:OFDM:SYMBol:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:OFDM:SYMBol:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.ofdm.symbol.
↪average.fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

return
evm_aver: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:OFDM:SYMBol:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.ofdm.symbol.
↪average.read(start = 1.0, count = 1.0, decimation = 1.0)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

return
evm_aver: No help available

6.4.1.9.2.46 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:OFDM:SYMBol:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:OFDM:SYMBol:CURRent
```

`class CurrentCls`

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:OFDM:SYMBol:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.ofdm.symbol.
↪current.fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

return
evm_curr: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:OFDM:SYMBol:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.ofdm.symbol.
↪current.read(start = 1.0, count = 1.0, decimation = 1.0)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

return
evm_curr: No help available

6.4.1.9.2.47 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:OFDM:SYMBol:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:OFDM:SYMBol:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:OFDM:SYMBol:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.ofdm.symbol.
↪maximum.fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

return
evm_max: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:OFDM:SYMBol:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.ofdm.symbol.
↪maximum.read(start = 1.0, count = 1.0, decimation = 1.0)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

return
evm_max: No help available

6.4.1.9.2.48 Symbol

class SymbolCls

Symbol commands group definition. 16 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.evMagnitude.symbol.clone()
```

Subgroups

6.4.1.9.2.49 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:SYMBol:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:SYMBol:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:SYMBol:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.symbol.average.
↪fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the EVM vs symbol traces for OFDM signals according to standard 802.11a, g, n, ac, ax, or be. The results of the current, average and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
evm_trace_aver: Comma-separated list of EVM values, one per OFDM symbol. The maximum number of values is 1366.

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:SYMBol:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.symbol.average.
↪read(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the EVM vs symbol traces for OFDM signals according to standard 802.11a, g, n, ac, ax, or be. The results of the current, average and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
evm_trace_aver: Comma-separated list of EVM values, one per OFDM symbol. The maximum number of values is 1366.

6.4.1.9.2.50 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:SYMBol:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:SYMBol:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:SYMBol:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.symbol.current.
↪fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the EVM vs symbol traces for OFDM signals according to standard 802.11a, g, n, ac, ax, or be. The results of the current, average and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
evm_trace_curr: Comma-separated list of EVM values, one per OFDM symbol. The maximum number of values is 1366.

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:SYMBol:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.symbol.current.
↪read(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the EVM vs symbol traces for OFDM signals according to standard 802.11a, g, n, ac, ax, or be. The results of the current, average and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
evm_trace_curr: Comma-separated list of EVM values, one per OFDM symbol. The maximum number of values is 1366.

6.4.1.9.2.51 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:SYMBol:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:SYMBol:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:SYMBol:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.symbol.maximum.
↪fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the EVM vs symbol traces for OFDM signals according to standard 802.11a, g, n, ac, ax, or be. The results of the current, average and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
evm_trace_max: Comma-separated list of EVM values, one per OFDM symbol. The maximum number of values is 1366.

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:SYMBol:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.symbol.maximum.
↪read(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the EVM vs symbol traces for OFDM signals according to standard 802.11a, g, n, ac, ax, or be. The results of the current, average and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
evm_trace_max: Comma-separated list of EVM values, one per OFDM symbol. The maximum number of values is 1366.

6.4.1.9.2.52 Mimo<Mimo>

RepCap Settings

```
# Range: Nr1 .. Nr8
rc = driver.wlanMeas.multiEval.trace.evMagnitude.symbol.mimo.repcap_mimo_get()
driver.wlanMeas.multiEval.trace.evMagnitude.symbol.mimo.repcap_mimo_set(repcap.Mimo.Nr1)
```

class MimoCls

Mimo commands group definition. 8 total commands, 4 Subgroups, 0 group commands Repeated Capability: Mimo, default value after init: Mimo.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.evMagnitude.symbol.mimo.clone()
```

Subgroups

6.4.1.9.2.53 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:SYMBol:MIMO<n>:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:SYMBol:MIMO<n>:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:SYMBol:MIMO<n>:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.symbol.mimo.
↪average.fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.
↪Default)
```

Return the values of the EVM vs symbol traces for signals for MIMO measurements per stream. The results of the current, average and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
evm_vs_sym_avg: Comma-separated list of EVM values, one per OFDM symbol. The maximum number of values is 1366.

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:SYMBol:MIMO<n>:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.symbol.mimo.
↪average.read(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.
↪Default)
```

Return the values of the EVM vs symbol traces for signals for MIMO measurements per stream. The results of the current, average and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
evm_vs_sym_avg: Comma-separated list of EVM values, one per OFDM symbol. The maximum number of values is 1366.

6.4.1.9.2.54 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:SYMBol:MIMO<n>:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:SYMBol:MIMO<n>:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:SYMBol:MIMO<n>:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.symbol.mimo.
↪current.fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.
↪Default)
```

Return the values of the EVM vs symbol traces for signals for MIMO measurements per stream. The results of the current, average and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
evm_vs_sym_cur: Comma-separated list of EVM values, one per OFDM symbol. The maximum number of values is 1366.

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:SYMBol:MIMO<n>:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.symbol.mimo.
↪current.read(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.
↪Default)
```

Return the values of the EVM vs symbol traces for signals for MIMO measurements per stream. The results of the current, average and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
evm_vs_sym_cur: Comma-separated list of EVM values, one per OFDM symbol. The maximum number of values is 1366.

6.4.1.9.2.55 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:SYMBol:MIMO<n>:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:SYMBol:MIMO<n>:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:SYMBol:MIMO<n>:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.symbol.mimo.
↪maximum.fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.
↪Default)
```

Return the values of the EVM vs symbol traces for signals for MIMO measurements per stream. The results of the current, average and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
evm_vs_sym_max: Comma-separated list of EVM values, one per OFDM symbol. The maximum number of values is 1366.

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:SYMBol:MIMO<n>:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.symbol.mimo.
↪maximum.read(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.
↪Default)
```

Return the values of the EVM vs symbol traces for signals for MIMO measurements per stream. The results of the current, average and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
evm_vs_sym_max: Comma-separated list of EVM values, one per OFDM symbol. The maximum number of values is 1366.

6.4.1.9.2.56 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:SYMBol:MIMO<n>:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:SYMBol:MIMO<n>:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:SYMBol:MIMO<n>:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.symbol.mimo.
```

(continues on next page)

(continued from previous page)

```
↪minimum.fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.
↪Default)
```

Return the values of the EVM vs symbol traces for signals for MIMO measurements per stream. The results of the current, average and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
evm_vs_sym_min: Comma-separated list of EVM values, one per OFDM symbol. The maximum number of values is 1366.

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:SYMBol:MIMO<n>:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.symbol.mimo.
↪minimum.read(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.
↪Default)
```

Return the values of the EVM vs symbol traces for signals for MIMO measurements per stream. The results of the current, average and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
evm_vs_sym_min: Comma-separated list of EVM values, one per OFDM symbol. The maximum number of values is 1366.

6.4.1.9.2.57 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:SYMBol:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:SYMBol:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:SYMBol:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.symbol.minimum.
↪fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the EVM vs symbol traces for OFDM signals according to standard 802.11a, g, n, ac, ax, or be. The results of the current, average and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
evm_trace_min: Comma-separated list of EVM values, one per OFDM symbol. The maximum number of values is 1366.

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:EVMagnitude:SYMBol:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.evMagnitude.symbol.minimum.
↪read(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the EVM vs symbol traces for OFDM signals according to standard 802.11a, g, n, ac, ax, or be. The results of the current, average and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
evm_trace_min: Comma-separated list of EVM values, one per OFDM symbol. The maximum number of values is 1366.

6.4.1.9.3 IqConstant

class IqConstantCls

IqConstant commands group definition. 4 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.iqConstant.clone()
```

Subgroups

6.4.1.9.3.1 Inphase

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:IQConst:INPHase
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:IQConst:INPHase
```

class InphaseCls

Inphase commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:IQConst:INPHase
value: List[float] = driver.wlanMeas.multiEval.trace.iqConstant.inphase.fetch()
```

Return the results in the I/Q constellation diagram. The I (in phase) and Q (quadrature) components are retrieved via separate commands.

Suppressed linked return values: reliability

return
iq_inphase: No help available

read() → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:IQConst:INPHase
value: List[float] = driver.wlanMeas.multiEval.trace.iqConstant.inphase.read()
```

Return the results in the I/Q constellation diagram. The I (in phase) and Q (quadrature) components are retrieved via separate commands.

Suppressed linked return values: reliability

return
iq_inphase: No help available

6.4.1.9.3.2 Quadrature

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:IQConst:QUADrature
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:IQConst:QUADrature
```

class QuadratureCls

Quadrature commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:IQConst:QUADrature
value: List[float] = driver.wlanMeas.multiEval.trace.iqConstant.quadrature.
↪fetch()
```

Return the results in the I/Q constellation diagram. The I (in phase) and Q (quadrature) components are retrieved via separate commands.

Suppressed linked return values: reliability

return
iq_quadrature: No help available

read() → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:IQConst:QUADrature
value: List[float] = driver.wlanMeas.multiEval.trace.iqConstant.quadrature.
↪read()
```

Return the results in the I/Q constellation diagram. The I (in phase) and Q (quadrature) components are retrieved via separate commands.

Suppressed linked return values: reliability

return
iq_quadrature: No help available

6.4.1.9.4 PowerVsTime

class PowerVsTimeCls

PowerVsTime commands group definition. 120 total commands, 9 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.powerVsTime.clone()
```

Subgroups

6.4.1.9.4.1 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.average.
↪fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the power vs time traces for SISO connections. The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
power_aver: Comma-separated list of max 1024 time values (1024 values without subarrays)

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.average.
↪read(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the power vs time traces for SISO connections. The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
power_aver: Comma-separated list of max 1024 time values (1024 values without subarrays)

6.4.1.9.4.2 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.current.
↪fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the power vs time traces for SISO connections. The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
power_curr: Comma-separated list of max 1024 time values (1024 values without subarrays)

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.current.
↪read(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the power vs time traces for SISO connections. The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
power_curr: Comma-separated list of max 1024 time values (1024 values without subarrays)

6.4.1.9.4.3 FallingEdge

class FallingEdgeCls

FallingEdge commands group definition. 40 total commands, 7 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.clone()
```

Subgroups

6.4.1.9.4.4 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.
↪average.fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the power vs time ramp traces, falling edge (FEDGe) and rising edge (REDGe) . The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
power_aver: Comma-separated list of power values For DSSS signals, the values are returned in percent relative to the configured reference power. For OFDM signals, the values are in dBm.

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.
↪average.read(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the power vs time ramp traces, falling edge (FEDGe) and rising edge (REDGe) . The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
power_aver: Comma-separated list of power values For DSSS signals, the values are returned in percent relative to the configured reference power. For OFDM signals, the values are in dBm.

6.4.1.9.4.5 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:CURRent
```

`class CurrentCls`

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

`fetch`(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.
↪current.fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the power vs time ramp traces, falling edge (FEDGe) and rising edge (REDGe) . The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
power_curr: Comma-separated list of power values For DSSS signals, the values are returned in percent relative to the configured reference power. For OFDM signals, the values are in dBm.

`read`(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.
↪current.read(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the power vs time ramp traces, falling edge (FEDGe) and rising edge (REDGe) . The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
power_curr: Comma-separated list of power values For DSSS signals, the values are returned in percent relative to the configured reference power. For OFDM signals, the values are in dBm.

6.4.1.9.4.6 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.
↪maximum.fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the power vs time ramp traces, falling edge (FEDGe) and rising edge (REDGe) . The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
power_max: Comma-separated list of power values For DSSS signals, the values are returned in percent relative to the configured reference power. For OFDM signals, the values are in dBm.

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.
↪maximum.read(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the power vs time ramp traces, falling edge (FEDGe) and rising edge (REDGe) . The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
power_max: Comma-separated list of power values For DSSS signals, the values are returned in percent relative to the configured reference power. For OFDM signals, the values are in dBm.

6.4.1.9.4.7 Mimo<Mimo>

RepCap Settings

```
# Range: Nr1 .. Nr8
rc = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.mimo.repcap_mimo_get()
driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.mimo.repcap_mimo_set(repcap.Mimo.
↪Nr1)
```

class MimoCls

Mimo commands group definition. 20 total commands, 6 Subgroups, 0 group commands Repeated Capability: Mimo, default value after init: Mimo.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.mimo.clone()
```

Subgroups

6.4.1.9.4.8 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MIMO<n>:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MIMO<n>:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decision: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MIMO<n>
↪:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.
↪mimo.average.fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.
↪Mimo.Default)
```

Return the values of the power vs time ramp traces for MIMO, rising edge (REDGe) and falling edge (FEDGe) . The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
power_aver: Comma-separated list of power values For DSSS signals, the values are returned in percent relative to the configured reference power. For OFDM signals, the values are in dBm.

read(*start: float = None*, *count: float = None*, *decision: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MIMO<n>
↪:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.
↪mimo.average.read(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.
↪Mimo.Default)
```

Return the values of the power vs time ramp traces for MIMO, rising edge (REDGe) and falling edge (FEDGe) . The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
power_aver: Comma-separated list of power values For DSSS signals, the values are returned in percent relative to the configured reference power. For OFDM signals, the values are in dBm.

6.4.1.9.4.9 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MIMO<n>:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MIMO<n>:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MIMO<n>
↪:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.
↪mimo.current.fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.
↪Mimo.Default)
```

Return the values of the power vs time ramp traces for MIMO, rising edge (REDGe) and falling edge (FEDGe) . The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
power_curr: Comma-separated list of power values For DSSS signals, the values are returned in percent relative to the configured reference power. For OFDM signals, the values are in dBm.

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MIMO<n>
↪:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.
↪mimo.current.read(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.
↪Mimo.Default)
```

Return the values of the power vs time ramp traces for MIMO, rising edge (REDGe) and falling edge (FEDGe) . The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
power_curr: Comma-separated list of power values For DSSS signals, the values are returned in percent relative to the configured reference power. For OFDM signals, the values are in dBm.

6.4.1.9.4.10 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MIMO<n>:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MIMO<n>:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MIMO<n>
↪:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.
↪mimo.maximum.fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.
↪Mimo.Default)
```

Return the values of the power vs time ramp traces for MIMO, rising edge (REDGe) and falling edge (FEDGe) . The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
power_max: Comma-separated list of power values For DSSS signals, the values are

returned in percent relative to the configured reference power. For OFDM signals, the values are in dBm.

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MIMO<n>
↪:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.
↪mimo.maximum.read(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.
↪Mimo.Default)
```

Return the values of the power vs time ramp traces for MIMO, rising edge (REDGe) and falling edge (FEDGe) . The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
power_max: Comma-separated list of power values For DSSS signals, the values are returned in percent relative to the configured reference power. For OFDM signals, the values are in dBm.

6.4.1.9.4.11 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MIMO<n>:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MIMO<n>:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MIMO<n>
↪:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.
↪mimo.minimum.fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.
↪Mimo.Default)
```

Return the values of the power vs time ramp traces for MIMO, rising edge (REDGe) and falling edge (FEDGe) . The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
power_min: Comma-separated list of power values For DSSS signals, the values are returned in percent relative to the configured reference power. For OFDM signals, the values are in dBm.

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MIMO<n>
↪:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.
↪mimo.minimum.read(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.
↪Mimo.Default)
```

Return the values of the power vs time ramp traces for MIMO, rising edge (REDGe) and falling edge (FEDGe) . The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
power_min: Comma-separated list of power values For DSSS signals, the values are returned in percent relative to the configured reference power. For OFDM signals, the values are in dBm.

6.4.1.9.4.12 Segment

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.mimo.segment.repcap_segment_
↪get()
driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.mimo.segment.repcap_segment_
↪set(repcap.Segment.Nr1)
```

class SegmentCls

Segment commands group definition. 10 total commands, 5 Subgroups, 0 group commands Repeated Capability: Segment, default value after init: Segment.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.mimo.segment.clone()
```

Subgroups

6.4.1.9.4.13 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MIMO<n>:SEGMent<seg>
↪:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MIMO<n>:SEGMent<seg>
↪:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MIMO<n>
↪:SEGMent<seg>:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.
↪mimo.segment.average.fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo =␣
↪repcap.Mimo.Default, segment = repcap.Segment.Default)
```

Return the values of the power vs time ramp traces, rising edge (REDGe) and falling edge (FEDGe) for MIMO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_aver: Comma-separated list of power values

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MIMO<n>
↪:SEGMent<seg>:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.
↪mimo.segment.average.read(start = 1.0, count = 1.0, decimation = 1.0, mimo =␣
↪repcap.Mimo.Default, segment = repcap.Segment.Default)
```

Return the values of the power vs time ramp traces, rising edge (REDGe) and falling edge (FEDGe) for MIMO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_aver: Comma-separated list of power values

6.4.1.9.4.14 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MIMO<n>:SEGMent<seg>
↪:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MIMO<n>:SEGMent<seg>
↪:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MIMO<n>
↪:SEGMent<seg>:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.
```

(continues on next page)

(continued from previous page)

```
↪mimo.segment.current.fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo =␣
↪repcap.Mimo.Default, segment = repcap.Segment.Default)
```

Return the values of the power vs time ramp traces, rising edge (REDGe) and falling edge (FEDGe) for MIMO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_curr: Comma-separated list of power values

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MIMO<n>
↪:SEGMent<seg>:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.
↪mimo.segment.current.read(start = 1.0, count = 1.0, decimation = 1.0, mimo =␣
↪repcap.Mimo.Default, segment = repcap.Segment.Default)
```

Return the values of the power vs time ramp traces, rising edge (REDGe) and falling edge (FEDGe) for MIMO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_curr: Comma-separated list of power values

6.4.1.9.4.15 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MIMO<n>:SEGMent<seg>
↪:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MIMO<n>:SEGMent<seg>
↪:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MIMO<n>
↪:SEGMent<seg>:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.
↪mimo.segment.maximum.fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo =␣
↪repcap.Mimo.Default, segment = repcap.Segment.Default)
```

Return the values of the power vs time ramp traces, rising edge (REDGe) and falling edge (FEDGe) for MIMO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_max: Comma-separated list of power values

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MIMO<n>
↪:SEGMent<seg>:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.
```

(continues on next page)

(continued from previous page)

```
↪mimo.segment.maximum.read(start = 1.0, count = 1.0, decimation = 1.0, mimo =␣
↪repcap.Mimo.Default, segment = repcap.Segment.Default)
```

Return the values of the power vs time ramp traces, rising edge (REDGe) and falling edge (FEDGe) for MIMO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_max: Comma-separated list of power values

6.4.1.9.4.16 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MIMO<n>:SEGMent<seg>
↪:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MIMO<n>:SEGMent<seg>
↪:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MIMO<n>
↪:SEGMent<seg>:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.
↪mimo.segment.minimum.fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo =␣
↪repcap.Mimo.Default, segment = repcap.Segment.Default)
```

Return the values of the power vs time ramp traces, rising edge (REDGe) and falling edge (FEDGe) for MIMO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_min: Comma-separated list of power values

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MIMO<n>
↪:SEGMent<seg>:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.
↪mimo.segment.minimum.read(start = 1.0, count = 1.0, decimation = 1.0, mimo =␣
↪repcap.Mimo.Default, segment = repcap.Segment.Default)
```

Return the values of the power vs time ramp traces, rising edge (REDGe) and falling edge (FEDGe) for MIMO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_min: Comma-separated list of power values

6.4.1.9.4.17 Time

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MIMO<n>:SEGMent<seg>:TIME
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MIMO<n>:SEGMent<seg>:TIME
```

class TimeCls

Time commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MIMO<n>
↪:SEGMent<seg>:TIME
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.
↪mimo.segment.time.fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo =␣
↪repcap.Mimo.Default, segment = repcap.Segment.Default)
```

Return the time indices for the power vs time ramp traces, rising edge (REDGe) and falling edge (FEDGe) for MIMO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. Refer to method RsCMPX_WlanMeas.WlanMeas.MultiEval.Trace.PowerVsTime.RisingEdge.Mimo.Segment.Current.fetch

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
time_values: Comma-separated list of time indices corresponding to the ramp power results.

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MIMO<n>
↪:SEGMent<seg>:TIME
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.
↪mimo.segment.time.read(start = 1.0, count = 1.0, decimation = 1.0, mimo =␣
↪repcap.Mimo.Default, segment = repcap.Segment.Default)
```

Return the time indices for the power vs time ramp traces, rising edge (REDGe) and falling edge (FEDGe) for MIMO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. Refer to method RsCMPX_WlanMeas.WlanMeas.MultiEval.Trace.PowerVsTime.RisingEdge.Mimo.Segment.Current.fetch

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
time_values: Comma-separated list of time indices corresponding to the ramp power results.

6.4.1.9.4.18 Time

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MIMO<n>:TIME
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MIMO<n>:TIME
```

class TimeCls

Time commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MIMO<n>
↪:TIME
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.
↪mimo.time.fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.
↪Mimo.Default)
```

Return the time indices for the power vs time ramp traces for MIMO, rising edge (REDGe) and falling edge (FEDGe) . Refer to method RsCMPX_WlanMeas.WlanMeas.MultiEval.Trace.PowerVsTime.FallingEdge.Mimo.Current.fetch etc.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
time_values: Comma-separated list of time indices corresponding to the ramp power results.

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MIMO<n>
↪:TIME
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.
↪mimo.time.read(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.
↪Default)
```

Return the time indices for the power vs time ramp traces for MIMO, rising edge (REDGe) and falling edge (FEDGe) . Refer to method RsCMPX_WlanMeas.WlanMeas.MultiEval.Trace.PowerVsTime.FallingEdge.Mimo.Current.fetch etc.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
time_values: Comma-separated list of time indices corresponding to the ramp power results.

6.4.1.9.4.19 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.
↪minimum.fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the power vs time ramp traces, falling edge (FEDGe) and rising edge (REDGe) . The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
power_min: Comma-separated list of power values For DSSS signals, the values are returned in percent relative to the configured reference power. For OFDM signals, the values are in dBm.

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.
↪minimum.read(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the power vs time ramp traces, falling edge (FEDGe) and rising edge (REDGe) . The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
power_min: Comma-separated list of power values For DSSS signals, the values are returned in percent relative to the configured reference power. For OFDM signals, the values are in dBm.

6.4.1.9.4.20 Segment

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.segment.repcap_segment_get()
driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.segment.repcap_segment_
↪set(repcap.Segment.Nr1)
```

class SegmentCls

Segment commands group definition. 10 total commands, 5 Subgroups, 0 group commands Repeated Capability: Segment, default value after init: Segment.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.segment.clone()
```

Subgroups

6.4.1.9.4.21 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:SEGMent<seg>:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:SEGMent<seg>:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:SEGMent
↪<seg>:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.
↪segment.average.fetch(start = 1.0, count = 1.0, decimation = 1.0, segment =␣
↪repcap.Segment.Default)
```

Return the values of the power vs time ramp traces, rising edge (REDGe) and falling edge (FEDGe) for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_aver: Comma-separated list of power values

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:SEGMent
↪<seg>:AVERage
```

(continues on next page)

(continued from previous page)

```
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.
↪segment.average.read(start = 1.0, count = 1.0, decimation = 1.0, segment =␣
↪repcap.Segment.Default)
```

Return the values of the power vs time ramp traces, rising edge (REDGe) and falling edge (FEDGe) for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_aver: Comma-separated list of power values

6.4.1.9.4.22 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:SEGMent<seg>:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:SEGMent<seg>:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:SEGMent
↪<seg>:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.
↪segment.current.fetch(start = 1.0, count = 1.0, decimation = 1.0, segment =␣
↪repcap.Segment.Default)
```

Return the values of the power vs time ramp traces, rising edge (REDGe) and falling edge (FEDGe) for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_curr: Comma-separated list of power values

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:SEGMent
↪<seg>:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.
↪segment.current.read(start = 1.0, count = 1.0, decimation = 1.0, segment =␣
↪repcap.Segment.Default)
```

Return the values of the power vs time ramp traces, rising edge (REDGe) and falling edge (FEDGe) for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_curr: Comma-separated list of power values

6.4.1.9.4.23 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:SEGMent<seg>:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:SEGMent<seg>:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:SEGMent
↪<seg>:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.
↪segment.maximum.fetch(start = 1.0, count = 1.0, decimation = 1.0, segment =␣
↪repcap.Segment.Default)
```

Return the values of the power vs time ramp traces, rising edge (REDGe) and falling edge (FEDGe) for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_max: Comma-separated list of power values

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:SEGMent
↪<seg>:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.
↪segment.maximum.read(start = 1.0, count = 1.0, decimation = 1.0, segment =␣
↪repcap.Segment.Default)
```

Return the values of the power vs time ramp traces, rising edge (REDGe) and falling edge (FEDGe) for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_max: Comma-separated list of power values

6.4.1.9.4.24 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:SEGMent<seg>:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:SEGMent<seg>:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:SEGMent
↪<seg>:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.
↪segment.minimum.fetch(start = 1.0, count = 1.0, decimation = 1.0, segment =␣
↪repcap.Segment.Default)
```

Return the values of the power vs time ramp traces, rising edge (REDGe) and falling edge (FEDGe) for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_min: Comma-separated list of power values

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:SEGMent
↪<seg>:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.
↪segment.minimum.read(start = 1.0, count = 1.0, decimation = 1.0, segment =␣
↪repcap.Segment.Default)
```

Return the values of the power vs time ramp traces, rising edge (REDGe) and falling edge (FEDGe) for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_min: Comma-separated list of power values

6.4.1.9.4.25 Time

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:SEGMent<seg>:TIME
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:SEGMent<seg>:TIME
```

class TimeCls

Time commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:SEGMent
↪<seg>:TIME
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.
↪segment.time.fetch(start = 1.0, count = 1.0, decimation = 1.0, segment =␣
↪repcap.Segment.Default)
```

Return the time indices for the power vs time ramp traces, rising edge (REDGe) and falling edge (FEDGe) for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. Refer to method RsCMPX_WlanMeas.WlanMeas.MultiEval.Trace.PowerVsTime.FallingEdge.Segment.Current.fetch etc.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
time_values: Comma-separated list of time indices corresponding to the ramp power results.

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:SEGMent
↪<seg>:TIME
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.
↪segment.time.read(start = 1.0, count = 1.0, decimation = 1.0, segment =␣
↪repcap.Segment.Default)
```

Return the time indices for the power vs time ramp traces, rising edge (REDGe) and falling edge (FEDGe) for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. Refer to method RsCMPX_WlanMeas.WlanMeas.MultiEval.Trace.PowerVsTime.FallingEdge.Segment.Current.fetch etc.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
time_values: Comma-separated list of time indices corresponding to the ramp power results.

6.4.1.9.4.26 Time

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:TIME
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:TIME
```

class TimeCls

Time commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:TIME
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.
↪time.fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the time indices corresponding to the power vs time ramp traces, falling edge (FEDGe) and rising edge (REDGe) . Refer to method RsCMPX_WlanMeas.WlanMeas.MultiEval.Trace.PowerVsTime.FallingEdge.Current.fetch etc.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
time_values: Comma-separated list of time indices corresponding to the ramp power results.

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:FEDGe:TIME
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.fallingEdge.
↪time.read(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the time indices corresponding to the power vs time ramp traces, falling edge (FEDGe) and rising edge (REDGe) . Refer to method RsCMPX_WlanMeas.WlanMeas.MultiEval.Trace.PowerVsTime.FallingEdge.Current.fetch etc.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
time_values: Comma-separated list of time indices corresponding to the ramp power results.

6.4.1.9.4.27 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.maximum.
↪fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the power vs time traces for SISO connections. The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
power_max: Comma-separated list of max 1024 time values (1024 values without subarrays)

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.maximum.
↪read(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the power vs time traces for SISO connections. The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
power_max: Comma-separated list of max 1024 time values (1024 values without subarrays)

6.4.1.9.4.28 Mimo<Mimo>

RepCap Settings

```
# Range: Nr1 .. Nr8
rc = driver.wlanMeas.multiEval.trace.powerVsTime.mimo.repcap_mimo_get()
driver.wlanMeas.multiEval.trace.powerVsTime.mimo.repcap_mimo_set(repcap.Mimo.Nr1)
```

class MimoCls

Mimo commands group definition. 20 total commands, 6 Subgroups, 0 group commands Repeated Capability: Mimo, default value after init: Mimo.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.powerVsTime.mimo.clone()
```

Subgroups

6.4.1.9.4.29 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MIMO<n>:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MIMO<n>:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MIMO<n>
↪:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.mimo.average.
↪fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.Default)
```

Return the values of the power vs time traces for MIMO. The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
power_aver: Comma-separated list of power values

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MIMO<n>:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.mimo.average.
↪read(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.Default)
```

Return the values of the power vs time traces for MIMO. The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
power_aver: Comma-separated list of power values

6.4.1.9.4.30 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MIMO<n>:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MIMO<n>:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MIMO<n>
↪:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.mimo.current.
↪fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.Default)
```

Return the values of the power vs time traces for MIMO. The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
power_curr: Comma-separated list of power values

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MIMO<n>:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.mimo.current.
↪read(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.Default)
```

Return the values of the power vs time traces for MIMO. The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
power_curr: Comma-separated list of power values

6.4.1.9.4.31 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MIMO<n>:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MIMO<n>:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MIMO<n>
↪:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.mimo.maximum.
↪fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.Default)
```

Return the values of the power vs time traces for MIMO. The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
power_max: Comma-separated list of power values

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MIMO<n>:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.mimo.maximum.
↪read(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.Default)
```

Return the values of the power vs time traces for MIMO. The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
power_max: Comma-separated list of power values

6.4.1.9.4.32 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MIMO<n>:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MIMO<n>:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MIMO<n>
↪:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.mimo.minimum.
↪fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.Default)
```

Return the values of the power vs time traces for MIMO. The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
power_min: Comma-separated list of power values

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MIMO<n>:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.mimo.minimum.
↪read(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.Default)
```

Return the values of the power vs time traces for MIMO. The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
power_min: Comma-separated list of power values

6.4.1.9.4.33 Segment

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.wlanMeas.multiEval.trace.powerVsTime.mimo.segment.repcap_segment_get()
driver.wlanMeas.multiEval.trace.powerVsTime.mimo.segment.repcap_segment_set(repcap.
↪Segment.Nr1)
```

class SegmentCls

Segment commands group definition. 10 total commands, 5 Subgroups, 0 group commands Repeated Capability: Segment, default value after init: Segment.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.powerVsTime.mimo.segment.clone()
```

Subgroups

6.4.1.9.4.34 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MIMO<n>:SEGMent<seg>:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MIMO<n>:SEGMent<seg>:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MIMO<n>
↪:SEGMent<seg>:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.mimo.segment.
↪average.fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.
↪Default, segment = repcap.Segment.Default)
```

Return the values of the power vs time traces for MIMO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_aver: Comma-separated list of max 1024 time values (1024 values without subarrays)

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MIMO<n>:SEGMent
↪<seg>:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.mimo.segment.
↪average.read(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.
↪Default, segment = repcap.Segment.Default)
```

Return the values of the power vs time traces for MIMO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_aver: Comma-separated list of max 1024 time values (1024 values without subarrays)

6.4.1.9.4.35 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MIMO<n>:SEGMent<seg>:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MIMO<n>:SEGMent<seg>:CURRent
```

`class CurrentCls`

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MIMO<n>
↪:SEGMent<seg>:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.mimo.segment.
↪current.fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.
↪Default, segment = repcap.Segment.Default)
```

Return the values of the power vs time traces for MIMO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_curr: Comma-separated list of max 1024 time values (1024 values without subarrays)

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MIMO<n>:SEGMent
↪<seg>:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.mimo.segment.
↪current.read(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.
↪Default, segment = repcap.Segment.Default)
```

Return the values of the power vs time traces for MIMO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_curr: Comma-separated list of max 1024 time values (1024 values without subarrays)

6.4.1.9.4.36 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MIMO<n>:SEGMent<seg>:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MIMO<n>:SEGMent<seg>:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MIMO<n>
↪:SEGMent<seg>:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.mimo.segment.
↪maximum.fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.
↪Default, segment = repcap.Segment.Default)
```

Return the values of the power vs time traces for MIMO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_max: Comma-separated list of max 1024 time values (1024 values without subarrays)

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MIMO<n>:SEGMent
↪<seg>:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.mimo.segment.
↪maximum.read(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.
↪Default, segment = repcap.Segment.Default)
```

Return the values of the power vs time traces for MIMO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_max: Comma-separated list of max 1024 time values (1024 values without subarrays)

6.4.1.9.4.37 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MIMO<n>:SEGMent<seg>:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MIMO<n>:SEGMent<seg>:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MIMO<n>
↪:SEGMent<seg>:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.mimo.segment.
↪minimum.fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.
↪Default, segment = repcap.Segment.Default)
```

Return the values of the power vs time traces for MIMO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_min: Comma-separated list of max 1024 time values (1024 values without subarrays)

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MIMO<n>:SEGMent
↪<seg>:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.mimo.segment.
↪minimum.read(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.
↪Default, segment = repcap.Segment.Default)
```

Return the values of the power vs time traces for MIMO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_min: Comma-separated list of max 1024 time values (1024 values without subarrays)

6.4.1.9.4.38 Time

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MIMO<n>:SEGMent<seg>:TIME
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MIMO<n>:SEGMent<seg>:TIME
```

class TimeCls

Time commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MIMO<n>
↪:SEGMent<seg>:TIME
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.mimo.segment.
↪time.fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.
↪Default, segment = repcap.Segment.Default)
```

Return the time indices for the current, average, minimum and maximum power vs time traces for MIMO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
time_values: Comma-separated list of max 1024 time values (1024 values without subarrays)

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MIMO<n>:SEGMent
↪<seg>:TIME
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.mimo.segment.
↪time.read(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.
↪Default, segment = repcap.Segment.Default)
```

Return the time indices for the current, average, minimum and maximum power vs time traces for MIMO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
time_values: Comma-separated list of max 1024 time values (1024 values without subarrays)

6.4.1.9.4.39 Time

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MIMO<n>:TIME
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MIMO<n>:TIME
```

class TimeCls

Time commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MIMO<n>:TIME
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.mimo.time.
↪fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.Default)
```

Return the time indices for the current, average, minimum and maximum power vs time traces (for MIMO) , see method RsCMPX_WlanMeas.WlanMeas.MultiEval.Trace.PowerVsTime.Mimo.Current.fetch etc.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
time_values: Comma-separated list of max 1024 time values (1024 values without subarrays)

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MIMO<n>:TIME
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.mimo.time.
↪read(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.Default)
```

Return the time indices for the current, average, minimum and maximum power vs time traces (for MIMO) , see method RsCMPX_WlanMeas.WlanMeas.MultiEval.Trace.PowerVsTime.Mimo.Current.fetch etc.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
time_values: Comma-separated list of max 1024 time values (1024 values without subarrays)

6.4.1.9.4.40 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.minimum.
↪fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the power vs time traces for SISO connections. The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
power_min: Comma-separated list of max 1024 time values (1024 values without subarrays)

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.minimum.
↪read(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the power vs time traces for SISO connections. The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
power_min: Comma-separated list of max 1024 time values (1024 values without subarrays)

6.4.1.9.4.41 RisingEdge

class RisingEdgeCls

RisingEdge commands group definition. 40 total commands, 7 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.clone()
```

Subgroups

6.4.1.9.4.42 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.
↪average.fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the power vs time ramp traces, falling edge (FEDGe) and rising edge (REDGe) . The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
power_aver: Comma-separated list of power values For DSSS signals, the values are returned in percent relative to the configured reference power. For OFDM signals, the values are in dBm.

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.
↪average.read(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the power vs time ramp traces, falling edge (FEDGe) and rising edge (REDGe) . The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
power_aver: Comma-separated list of power values For DSSS signals, the values are returned in percent relative to the configured reference power. For OFDM signals, the values are in dBm.

6.4.1.9.4.43 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.
↪current.fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the power vs time ramp traces, falling edge (FEDGe) and rising edge (REDGe) . The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
power_curr: Comma-separated list of power values For DSSS signals, the values are returned in percent relative to the configured reference power. For OFDM signals, the values are in dBm.

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.
↪current.read(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the power vs time ramp traces, falling edge (FEDGe) and rising edge (REDGe) . The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
power_curr: Comma-separated list of power values For DSSS signals, the values are returned in percent relative to the configured reference power. For OFDM signals, the values are in dBm.

6.4.1.9.4.44 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.
↪maximum.fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the power vs time ramp traces, falling edge (FEDGe) and rising edge (REDGe) . The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
power_max: Comma-separated list of power values For DSSS signals, the values are returned in percent relative to the configured reference power. For OFDM signals, the values are in dBm.

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.
↪maximum.read(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the power vs time ramp traces, falling edge (FEDGe) and rising edge (REDGe) . The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
power_max: Comma-separated list of power values For DSSS signals, the values are returned in percent relative to the configured reference power. For OFDM signals, the values are in dBm.

6.4.1.9.4.45 Mimo<Mimo>

RepCap Settings

```
# Range: Nr1 .. Nr8
rc = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.mimo.repcap_mimo_get()
driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.mimo.repcap_mimo_set(repcap.Mimo.
↪Nr1)
```

class MimoCls

Mimo commands group definition. 20 total commands, 6 Subgroups, 0 group commands Repeated Capability: Mimo, default value after init: Mimo.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.mimo.clone()
```

Subgroups

6.4.1.9.4.46 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MIMO<n>:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MIMO<n>:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MIMO<n>
↪:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.
↪mimo.average.fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.
↪Mimo.Default)
```

Return the values of the power vs time ramp traces for MIMO, rising edge (REDGe) and falling edge (FEDGe) . The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
power_aver: Comma-separated list of power values For DSSS signals, the values are returned in percent relative to the configured reference power. For OFDM signals, the values are in dBm.

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MIMO<n>
↪:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.
↪mimo.average.read(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.
↪Mimo.Default)
```

Return the values of the power vs time ramp traces for MIMO, rising edge (REDGe) and falling edge (FEDGe) . The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
power_aver: Comma-separated list of power values For DSSS signals, the values are returned in percent relative to the configured reference power. For OFDM signals, the values are in dBm.

6.4.1.9.4.47 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MIMO<n>:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MIMO<n>:CURRent
```

`class CurrentCls`

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MIMO<n>
↪:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.
↪mimo.current.fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.
↪Mimo.Default)
```

Return the values of the power vs time ramp traces for MIMO, rising edge (REDGe) and falling edge (FEDGe) . The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
power_curr: Comma-separated list of power values For DSSS signals, the values are returned in percent relative to the configured reference power. For OFDM signals, the values are in dBm.

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MIMO<n>
↪:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.
↪mimo.current.read(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.
↪Mimo.Default)
```

Return the values of the power vs time ramp traces for MIMO, rising edge (REDGe) and falling edge (FEDGe) . The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
power_curr: Comma-separated list of power values For DSSS signals, the values are returned in percent relative to the configured reference power. For OFDM signals, the values are in dBm.

6.4.1.9.4.48 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MIMO<n>:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MIMO<n>:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MIMO<n>
↪:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.
↪mimo.maximum.fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.
↪Mimo.Default)
```

Return the values of the power vs time ramp traces for MIMO, rising edge (REDGe) and falling edge (FEDGe) . The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
power_max: Comma-separated list of power values For DSSS signals, the values are returned in percent relative to the configured reference power. For OFDM signals, the values are in dBm.

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MIMO<n>
↪:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.
↪mimo.maximum.read(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.
↪Mimo.Default)
```

Return the values of the power vs time ramp traces for MIMO, rising edge (REDGe) and falling edge (FEDGe) . The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
power_max: Comma-separated list of power values For DSSS signals, the values are returned in percent relative to the configured reference power. For OFDM signals, the values are in dBm.

6.4.1.9.4.49 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MIMO<n>:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MIMO<n>:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MIMO<n>
↪:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.
↪mimo.minimum.fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.
↪Mimo.Default)
```

Return the values of the power vs time ramp traces for MIMO, rising edge (REDGe) and falling edge (FEDGe) . The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
power_min: Comma-separated list of power values For DSSS signals, the values are returned in percent relative to the configured reference power. For OFDM signals, the values are in dBm.

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MIMO<n>
↪:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.
↪mimo.minimum.read(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.
↪Mimo.Default)
```

Return the values of the power vs time ramp traces for MIMO, rising edge (REDGe) and falling edge (FEDGe) . The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
power_min: Comma-separated list of power values For DSSS signals, the values are returned in percent relative to the configured reference power. For OFDM signals, the values are in dBm.

6.4.1.9.4.50 Segment

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.mimo.segment.repcap_segment_
↪get()
driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.mimo.segment.repcap_segment_
↪set(repcap.Segment.Nr1)
```

class SegmentCls

Segment commands group definition. 10 total commands, 5 Subgroups, 0 group commands Repeated Capability: Segment, default value after init: Segment.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.mimo.segment.clone()
```

Subgroups

6.4.1.9.4.51 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MIMO<n>:SEGMent<seg>
↪:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MIMO<n>:SEGMent<seg>
↪:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MIMO<n>
↪:SEGMent<seg>:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.
↪mimo.segment.average.fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo =␣
↪repcap.Mimo.Default, segment = repcap.Segment.Default)
```

Return the values of the power vs time ramp traces, rising edge (REDGe) and falling edge (FEDGe) for MIMO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_aver: Comma-separated list of power values

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MIMO<n>
↪:SEGMent<seg>:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.
↪mimo.segment.average.read(start = 1.0, count = 1.0, decimation = 1.0, mimo =␣
↪repcap.Mimo.Default, segment = repcap.Segment.Default)
```

Return the values of the power vs time ramp traces, rising edge (REDGe) and falling edge (FEDGe) for MIMO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_aver: Comma-separated list of power values

6.4.1.9.4.52 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MIMO<n>:SEGMent<seg>
↪:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MIMO<n>:SEGMent<seg>
↪:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MIMO<n>
↪:SEGMent<seg>:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.
↪mimo.segment.current.fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo =␣
↪repcap.Mimo.Default, segment = repcap.Segment.Default)
```

Return the values of the power vs time ramp traces, rising edge (REDGe) and falling edge (FEDGe) for MIMO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_curr: Comma-separated list of power values

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MIMO<n>
↪:SEGMent<seg>:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.
↪mimo.segment.current.read(start = 1.0, count = 1.0, decimation = 1.0, mimo =␣
↪repcap.Mimo.Default, segment = repcap.Segment.Default)
```

Return the values of the power vs time ramp traces, rising edge (REDGe) and falling edge (FEDGe) for MIMO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_curr: Comma-separated list of power values

6.4.1.9.4.53 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MIMO<n>:SEGMent<seg>
↪:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MIMO<n>:SEGMent<seg>
↪:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MIMO<n>
↪:SEGMent<seg>:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.
↪mimo.segment.maximum.fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo =␣
↪repcap.Mimo.Default, segment = repcap.Segment.Default)
```

Return the values of the power vs time ramp traces, rising edge (REDGe) and falling edge (FEDGe) for MIMO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_max: Comma-separated list of power values

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MIMO<n>
↪:SEGMent<seg>:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.
↪mimo.segment.maximum.read(start = 1.0, count = 1.0, decimation = 1.0, mimo =␣
↪repcap.Mimo.Default, segment = repcap.Segment.Default)
```

Return the values of the power vs time ramp traces, rising edge (REDGe) and falling edge (FEDGe) for MIMO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_max: Comma-separated list of power values

6.4.1.9.4.54 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MIMO<n>:SEGMent<seg>
↪:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MIMO<n>:SEGMent<seg>
↪:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MIMO<n>
↪:SEGMent<seg>:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.
↪mimo.segment.minimum.fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo =␣
↪repcap.Mimo.Default, segment = repcap.Segment.Default)
```

Return the values of the power vs time ramp traces, rising edge (REDGe) and falling edge (FEDGe) for MIMO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_min: Comma-separated list of power values

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MIMO<n>
↪:SEGMent<seg>:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.
↪mimo.segment.minimum.read(start = 1.0, count = 1.0, decimation = 1.0, mimo =␣
↪repcap.Mimo.Default, segment = repcap.Segment.Default)
```

Return the values of the power vs time ramp traces, rising edge (REDGe) and falling edge (FEDGe) for MIMO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_min: Comma-separated list of power values

6.4.1.9.4.55 Time

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MIMO<n>:SEGMent<seg>:TIME
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MIMO<n>:SEGMent<seg>:TIME
```

class TimeCls

Time commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MIMO<n>
↪:SEGMent<seg>:TIME
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.
↪mimo.segment.time.fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo =␣
↪repcap.Mimo.Default, segment = repcap.Segment.Default)
```

Return the time indices for the power vs time ramp traces, rising edge (REDGe) and falling edge (FEDGe) for MIMO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. Refer to method RsCMPX_WlanMeas.WlanMeas.MultiEval.Trace.PowerVsTime.RisingEdge.Mimo.Segment.Current.fetch

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
time_values: Comma-separated list of time indices corresponding to the ramp power results.

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MIMO<n>
↪:SEGMent<seg>:TIME
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.
↪mimo.segment.time.read(start = 1.0, count = 1.0, decimation = 1.0, mimo =␣
↪repcap.Mimo.Default, segment = repcap.Segment.Default)
```

Return the time indices for the power vs time ramp traces, rising edge (REDGe) and falling edge (FEDGe) for MIMO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. Refer to method RsCMPX_WlanMeas.WlanMeas.MultiEval.Trace.PowerVsTime.RisingEdge.Mimo.Segment.Current.fetch

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
time_values: Comma-separated list of time indices corresponding to the ramp power results.

6.4.1.9.4.56 Time

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MIMO<n>:TIME
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MIMO<n>:TIME
```

class TimeCls

Time commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MIMO<n>
↪:TIME
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.
↪mimo.time.fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.
↪Mimo.Default)
```

Return the time indices for the power vs time ramp traces for MIMO, rising edge (REDGe) and falling edge (FEDGe) . Refer to method RsCMPX_WlanMeas.WlanMeas.MultiEval.Trace.PowerVsTime.FallingEdge.Mimo.Current.fetch etc.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
time_values: Comma-separated list of time indices corresponding to the ramp power results.

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MIMO<n>
↪:TIME
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.
↪mimo.time.read(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.
↪Default)
```

Return the time indices for the power vs time ramp traces for MIMO, rising edge (REDGe) and falling edge (FEDGe) . Refer to method RsCMPX_WlanMeas.WlanMeas.MultiEval.Trace.PowerVsTime.FallingEdge.Mimo.Current.fetch etc.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
time_values: Comma-separated list of time indices corresponding to the ramp power results.

6.4.1.9.4.57 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.
↪minimum.fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the power vs time ramp traces, falling edge (FEDGe) and rising edge (REDGe) . The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
power_min: Comma-separated list of power values For DSSS signals, the values are returned in percent relative to the configured reference power. For OFDM signals, the values are in dBm.

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.
↪minimum.read(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the power vs time ramp traces, falling edge (FEDGe) and rising edge (REDGe) . The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
power_min: Comma-separated list of power values For DSSS signals, the values are returned in percent relative to the configured reference power. For OFDM signals, the values are in dBm.

6.4.1.9.4.58 Segment

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.segment.repcap_segment_get()
driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.segment.repcap_segment_set(repcap.
↪Segment.Nr1)
```

class SegmentCls

Segment commands group definition. 10 total commands, 5 Subgroups, 0 group commands Repeated Capability: Segment, default value after init: Segment.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.segment.clone()
```

Subgroups

6.4.1.9.4.59 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:SEGMent<seg>:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:SEGMent<seg>:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:SEGMent
↪<seg>:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.
↪segment.average.fetch(start = 1.0, count = 1.0, decimation = 1.0, segment =␣
↪repcap.Segment.Default)
```

Return the values of the power vs time ramp traces, rising edge (REDGe) and falling edge (FEDGe) for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_aver: Comma-separated list of power values

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:SEGMent
↪<seg>:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.
↪segment.average.read(start = 1.0, count = 1.0, decimation = 1.0, segment =␣
↪repcap.Segment.Default)
```

Return the values of the power vs time ramp traces, rising edge (REDGe) and falling edge (FEDGe) for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_aver: Comma-separated list of power values

6.4.1.9.4.60 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:SEGMent<seg>:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:SEGMent<seg>:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:SEGMent
↪<seg>:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.
↪segment.current.fetch(start = 1.0, count = 1.0, decimation = 1.0, segment =␣
↪repcap.Segment.Default)
```

Return the values of the power vs time ramp traces, rising edge (REDGe) and falling edge (FEDGe) for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_curr: Comma-separated list of power values

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:SEGMent
↪<seg>:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.
↪segment.current.read(start = 1.0, count = 1.0, decimation = 1.0, segment =␣
↪repcap.Segment.Default)
```

Return the values of the power vs time ramp traces, rising edge (REDGe) and falling edge (FEDGe) for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_curr: Comma-separated list of power values

6.4.1.9.4.61 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:SEGMent<seg>:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:SEGMent<seg>:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:SEGMent
↪<seg>:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.
↪segment.maximum.fetch(start = 1.0, count = 1.0, decimation = 1.0, segment =␣
↪repcap.Segment.Default)
```

Return the values of the power vs time ramp traces, rising edge (REDGe) and falling edge (FEDGe) for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_max: Comma-separated list of power values

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:SEGMent
↪<seg>:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.
↪segment.maximum.read(start = 1.0, count = 1.0, decimation = 1.0, segment =␣
↪repcap.Segment.Default)
```

Return the values of the power vs time ramp traces, rising edge (REDGe) and falling edge (FEDGe) for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_max: Comma-separated list of power values

6.4.1.9.4.62 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:SEGMent<seg>:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:SEGMent<seg>:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:SEGMent
↪<seg>:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.
↪segment.minimum.fetch(start = 1.0, count = 1.0, decimation = 1.0, segment =␣
↪repcap.Segment.Default)
```

Return the values of the power vs time ramp traces, rising edge (REDGe) and falling edge (FEDGe) for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_min: Comma-separated list of power values

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:SEGMent
↪<seg>:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.
↪segment.minimum.read(start = 1.0, count = 1.0, decimation = 1.0, segment =␣
↪repcap.Segment.Default)
```

Return the values of the power vs time ramp traces, rising edge (REDGe) and falling edge (FEDGe) for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_min: Comma-separated list of power values

6.4.1.9.4.63 Time

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:SEGMent<seg>:TIME
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:SEGMent<seg>:TIME
```

class TimeCls

Time commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:SEGMent
↪<seg>:TIME
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.
↪segment.time.fetch(start = 1.0, count = 1.0, decimation = 1.0, segment =␣
↪repcap.Segment.Default)
```

Return the time indices for the power vs time ramp traces, rising edge (REDGe) and falling edge (FEDGe) for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. Refer to method RsCMPX_WlanMeas.WlanMeas.MultiEval.Trace.PowerVsTime.FallingEdge.Segment.Current.fetch etc.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
time_values: Comma-separated list of time indices corresponding to the ramp power results.

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:SEGMent
↪<seg>:TIME
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.
↪segment.time.read(start = 1.0, count = 1.0, decimation = 1.0, segment =␣
↪repcap.Segment.Default)
```

Return the time indices for the power vs time ramp traces, rising edge (REDGe) and falling edge (FEDGe) for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. Refer to method RsCMPX_WlanMeas.WlanMeas.MultiEval.Trace.PowerVsTime.FallingEdge.Segment.Current.fetch etc.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
time_values: Comma-separated list of time indices corresponding to the ramp power results.

6.4.1.9.4.64 Time

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:TIME
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:TIME
```

class TimeCls

Time commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:TIME
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.
↪time.fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the time indices corresponding to the power vs time ramp traces, falling edge (FEDGe) and rising edge (REDGe) . Refer to method RsCMPX_WlanMeas.WlanMeas.MultiEval.Trace.PowerVsTime.FallingEdge.Current.fetch etc.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
time_values: Comma-separated list of time indices corresponding to the ramp power results.

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:REDGe:TIME
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.risingEdge.
↪time.read(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the time indices corresponding to the power vs time ramp traces, falling edge (FEDGe) and rising edge (REDGe) . Refer to method RsCMPX_WlanMeas.WlanMeas.MultiEval.Trace.PowerVsTime.FallingEdge.Current.fetch etc.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
time_values: Comma-separated list of time indices corresponding to the ramp power results.

6.4.1.9.4.65 Segment

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.wlanMeas.multiEval.trace.powerVsTime.segment.repcap_segment_get()
driver.wlanMeas.multiEval.trace.powerVsTime.segment.repcap_segment_set(repcap.Segment.
↪Nr1)
```

class SegmentCls

Segment commands group definition. 10 total commands, 5 Subgroups, 0 group commands Repeated Capability: Segment, default value after init: Segment.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.powerVsTime.segment.clone()
```

Subgroups

6.4.1.9.4.66 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:SEGMent<seg>:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:SEGMent<seg>:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:SEGMent<seg>
↪:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.segment.
↪average.fetch(start = 1.0, count = 1.0, decimation = 1.0, segment = repcap.
↪Segment.Default)
```

Return the values of the power vs time traces for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_aver: Comma-separated list of power values

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:SEGMent<seg>
↪:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.segment.
↪average.read(start = 1.0, count = 1.0, decimation = 1.0, segment = repcap.
↪Segment.Default)
```

Return the values of the power vs time traces for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_aver: Comma-separated list of power values

6.4.1.9.4.67 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:SEGMent<seg>:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:SEGMent<seg>:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:SEGMent<seg>
↪:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.segment.
↪current.fetch(start = 1.0, count = 1.0, decimation = 1.0, segment = repcap.
↪Segment.Default)
```

Return the values of the power vs time traces for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_curr: Comma-separated list of power values

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:SEGMent<seg>
↪:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.segment.
↪current.read(start = 1.0, count = 1.0, decimation = 1.0, segment = repcap.
↪Segment.Default)
```

Return the values of the power vs time traces for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_curr: Comma-separated list of power values

6.4.1.9.4.68 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:SEGMent<seg>:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:SEGMent<seg>:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:SEGMent<seg>
↪:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.segment.
↪maximum.fetch(start = 1.0, count = 1.0, decimation = 1.0, segment = repcap.
↪Segment.Default)
```

Return the values of the power vs time traces for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_max: Comma-separated list of power values

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:SEGMent<seg>
↪:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.segment.
↪maximum.read(start = 1.0, count = 1.0, decimation = 1.0, segment = repcap.
↪Segment.Default)
```

Return the values of the power vs time traces for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_max: Comma-separated list of power values

6.4.1.9.4.69 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:SEGMent<seg>:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:SEGMent<seg>:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:SEGMent<seg>
↪:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.segment.
↪minimum.fetch(start = 1.0, count = 1.0, decimation = 1.0, segment = repcap.
↪Segment.Default)
```

Return the values of the power vs time traces for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_min: Comma-separated list of power values

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:SEGMent<seg>
↪:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.segment.
↪minimum.read(start = 1.0, count = 1.0, decimation = 1.0, segment = repcap.
↪Segment.Default)
```

Return the values of the power vs time traces for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. The results of the current, average, maximum and minimum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
power_min: Comma-separated list of power values

6.4.1.9.4.70 Time

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:SEGMent<seg>:TIME
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:SEGMent<seg>:TIME
```

class TimeCls

Time commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:SEGMent<seg>
↪:TIME
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.segment.time.
↪fetch(start = 1.0, count = 1.0, decimation = 1.0, segment = repcap.Segment.
↪Default)
```

Return the time indices for the current, average, minimum and maximum power vs time traces for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. See also method RsCMPX_WlanMeas.WlanMeas.MultiEval.Trace.PowerVsTime.Segment.Current.fetch etc.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
time_values: Comma-separated list of max 1024 time values (1024 values without subarrays)

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:SEGMent<seg>
↪:TIME
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.segment.time.
↪read(start = 1.0, count = 1.0, decimation = 1.0, segment = repcap.Segment.
↪Default)
```

Return the time indices for the current, average, minimum and maximum power vs time traces for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. See also method RsCMPX_WlanMeas.WlanMeas.MultiEval.Trace.PowerVsTime.Segment.Current.fetch etc.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
time_values: Comma-separated list of max 1024 time values (1024 values without subarrays)

6.4.1.9.4.71 Time

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:TIME
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:TIME
```

class TimeCls

Time commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:TIME
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.time.
↪fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the time indices for the current, average, minimum and maximum power vs time traces, see method RsCMPX_WlanMeas. WlanMeas.MultiEval.Trace.PowerVsTime.Current.fetch etc.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
time_values: Comma-separated list of max 1024 time values (1024 values without subarrays)

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:TIME
value: List[float] = driver.wlanMeas.multiEval.trace.powerVsTime.time.
↪read(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the time indices for the current, average, minimum and maximum power vs time traces, see method RsCMPX_WlanMeas. WlanMeas.MultiEval.Trace.PowerVsTime.Current.fetch etc.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
time_values: Comma-separated list of max 1024 time values (1024 values without subarrays)

6.4.1.9.5 SpectrFlatness

class SpectrFlatnessCls

SpectrFlatness commands group definition. 104 total commands, 9 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.spectrFlatness.clone()
```

Subgroups

6.4.1.9.5.1 Acarrier

class AcarrierCls

Acarrier commands group definition. 24 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.spectrFlatness.acarrier.clone()
```

Subgroups

6.4.1.9.5.2 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:ACARrier:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:ACARrier:AVERage
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:ACARrier:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:ACARrier:AVERage
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.trace.
↪spectrFlatness.acarrier.average.calculate(start = 1.0, count = 1.0,␣
↪decimation = 1.0)
```

Return the results over all carriers (complete FFTSize) of the spectrum flatness traces for OFDM and OFDMA SISO signals. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
sflat_all_carr_avg: Comma-separated list of power levels, one value per subcarrier (including data, pilot and unused subcarriers) The number of power levels depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:ACARrier:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.acarrier.
↪average.fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the results over all carriers (complete FFTSize) of the spectrum flatness traces for OFDM and OFDMA SISO signals. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
sflat_all_carr_avg: Comma-separated list of power levels, one value per subcarrier (including data, pilot and unused subcarriers) The number of power levels depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:ACARrier:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.acarrier.
↪average.read(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the results over all carriers (complete FFTSize) of the spectrum flatness traces for OFDM and OFDMA SISO signals. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
sflat_all_carr_avg: Comma-separated list of power levels, one value per subcarrier (including data, pilot and unused subcarriers) The number of power levels depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

6.4.1.9.5.3 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:ACARrier:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:ACARrier:CURRent
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:ACARrier:CURRent
```

class CurrentCls

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:ACARrier:CURRent
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.trace.
↪spectrFlatness.acarrier.current.calculate(start = 1.0, count = 1.0,␣
↪decimation = 1.0)
```

Return the results over all carriers (complete FFTSize) of the spectrum flatness traces for OFDM and OFDMA SISO signals. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
sflat_all_carr_cur: Comma-separated list of power levels, one value per subcarrier (including data, pilot and unused subcarriers) The number of power levels depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:ACARrier:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.acarrier.
↪current.fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the results over all carriers (complete FFTSize) of the spectrum flatness traces for OFDM and OFDMA SISO signals. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
sflat_all_carr_cur: Comma-separated list of power levels, one value per subcarrier (including data, pilot and unused subcarriers) The number of power levels depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:ACARrier:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.acarrier.
↪current.read(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the results over all carriers (complete FFTSize) of the spectrum flatness traces for OFDM and OFDMA SISO signals. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
sflat_all_carr_cur: Comma-separated list of power levels, one value per subcarrier (including data, pilot and unused subcarriers) The number of power levels depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

6.4.1.9.5.4 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:ACARrier:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:ACARrier:MAXimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:ACARrier:MAXimum
```

class MaximumCls

Maximum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:ACARrier:MAXimum
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.trace.
↪spectrFlatness.acarrier.maximum.calculate(start = 1.0, count = 1.0,␣
↪decimation = 1.0)
```

Return the results over all carriers (complete FFTSize) of the spectrum flatness traces for OFDM and OFDMA SISO signals. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
sflat_all_carr_max: Comma-separated list of power levels, one value per subcarrier (including data, pilot and unused subcarriers) The number of power levels depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:ACARrier:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.acarrier.
↪maximum.fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the results over all carriers (complete FFTSize) of the spectrum flatness traces for OFDM and OFDMA SISO signals. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
sflat_all_carr_max: Comma-separated list of power levels, one value per subcarrier (including data, pilot and unused subcarriers) The number of power levels depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:ACARrier:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.acarrier.
↪maximum.read(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the results over all carriers (complete FFTSize) of the spectrum flatness traces for OFDM and OFDMA SISO signals. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
sflat_all_carr_max: Comma-separated list of power levels, one value per subcarrier (including data, pilot and unused subcarriers) The number of power levels depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

6.4.1.9.5.5 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:ACARrier:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:ACARrier:MINimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:ACARrier:MINimum
```

class MinimumCls

Minimum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:ACARrier:MINimum
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.trace.
↪spectrFlatness.acarrier.minimum.calculate(start = 1.0, count = 1.0,␣
↪decimation = 1.0)
```

Return the results over all carriers (complete FFTSize) of the spectrum flatness traces for OFDM and OFDMA SISO signals. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
sflat_all_carr_min: Comma-separated list of power levels, one value per subcarrier (including data, pilot and unused subcarriers) The number of power levels depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:ACARrier:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.acarrier.
↪minimum.fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the results over all carriers (complete FFTSize) of the spectrum flatness traces for OFDM and OFDMA SISO signals. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
sflat_all_carr_min: Comma-separated list of power levels, one value per subcarrier (including data, pilot and unused subcarriers) The number of power levels depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:ACARrier:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.acarrier.
↪minimum.read(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the results over all carriers (complete FFTSize) of the spectrum flatness traces for OFDM and OFDMA SISO signals. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
sflat_all_carr_min: Comma-separated list of power levels, one value per subcarrier (including data, pilot and unused subcarriers) The number of power levels depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

6.4.1.9.5.6 Segment

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.wlanMeas.multiEval.trace.spectrFlatness.acarrier.segment.repcap_segment_get()
driver.wlanMeas.multiEval.trace.spectrFlatness.acarrier.segment.repcap_segment_
↪set(repcap.Segment.Nr1)
```

class SegmentCls

Segment commands group definition. 12 total commands, 4 Subgroups, 0 group commands Repeated Capability: Segment, default value after init: Segment.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.spectrFlatness.acarrier.segment.clone()
```

Subgroups

6.4.1.9.5.7 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:ACARrier:SEGMent<seg>:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:ACARrier:SEGMent<seg>
↪:AVERage
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:ACARrier:SEGMent<seg>
↪:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:ACARrier:SEGMent<seg>:AVERage
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.trace.
↪spectrFlatness.acarrier.segment.average.calculate(start = 1.0, count = 1.0,␣
↪decimation = 1.0, segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
sflat_all_carr_avg: No help available

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:ACARrier:SEGMent<seg>:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.acarrier.
↪segment.average.fetch(start = 1.0, count = 1.0, decimation = 1.0, segment =␣
↪repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
sflat_all_carr_avg: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:ACARrier:SEGMent<seg>:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.acarrier.
↪segment.average.read(start = 1.0, count = 1.0, decimation = 1.0, segment =␣
↪repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
sflat_all_carr_avg: No help available

6.4.1.9.5.8 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:ACARrier:SEGMent<seg>:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:ACARrier:SEGMent<seg>
↪:CURRent
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:ACARrier:SEGMent<seg>
↪:CURRent
```

class CurrentCls

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:ACARrier:SEGMent<seg>:CURRent
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.trace.
↪spectrFlatness.acarrier.segment.current.calculate(start = 1.0, count = 1.0,␣
↪decimation = 1.0, segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
sflat_all_carr_cur: No help available

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:ACARrier:SEGMent<seg>:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.acarrier.
```

(continues on next page)

(continued from previous page)

```
↪segment.current.fetch(start = 1.0, count = 1.0, decimation = 1.0, segment =␣
↪repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
sflat_all_carr_cur: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:ACARrier:SEGMent<seg>:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.acarrier.
↪segment.current.read(start = 1.0, count = 1.0, decimation = 1.0, segment =␣
↪repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
sflat_all_carr_cur: No help available

6.4.1.9.5.9 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:ACARrier:SEGMent<seg>:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:ACARrier:SEGMent<seg>
↪:MAXimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:ACARrier:SEGMent<seg>
↪:MAXimum
```

class MaximumCls

Maximum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:ACARrier:SEGMent<seg>:MAXimum
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.trace.
↪spectrFlatness.acarrier.segment.maximum.calculate(start = 1.0, count = 1.0,␣
↪decimation = 1.0, segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
sflat_all_carr_max: No help available

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:ACARrier:SEGMent<seg>:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.acarrier.
↪segment.maximum.fetch(start = 1.0, count = 1.0, decimation = 1.0, segment =␣
↪repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
sflat_all_carr_max: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:ACARrier:SEGMent<seg>:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.acarrier.
↪segment.maximum.read(start = 1.0, count = 1.0, decimation = 1.0, segment =␣
↪repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
sflat_all_carr_max: No help available

6.4.1.9.5.10 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:ACARrier:SEGMent<seg>:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:ACARrier:SEGMent<seg>
↪:MINimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:ACARrier:SEGMent<seg>
↪:MINimum
```

class MinimumCls

Minimum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:ACARrier:SEGMent<seg>:MINimum
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.trace.
↪spectrFlatness.acarrier.segment.minimum.calculate(start = 1.0, count = 1.0,␣
↪decimation = 1.0, segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
sflat_all_carr_min: No help available

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:ACARrier:SEGMent<seg>:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.acarrier.
↪segment.minimum.fetch(start = 1.0, count = 1.0, decimation = 1.0, segment =␣
↪repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
sflat_all_carr_min: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:ACARrier:SEGMent<seg>:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.acarrier.
↪segment.minimum.read(start = 1.0, count = 1.0, decimation = 1.0, segment =␣
↪repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
sflat_all_carr_min: No help available

6.4.1.9.5.11 Acsiso

class AcsisoCls

Acsiso commands group definition. 8 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.spectrFlatness.acsiso.clone()
```

Subgroups

6.4.1.9.5.12 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:ACSiso:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:ACSiso:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:ACSiso:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.acsiso.
↪average.fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

return
sflat_aver: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:ACSiso:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.acsiso.
↪average.read(start = 1.0, count = 1.0, decimation = 1.0)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

return
sflat_aver: No help available

6.4.1.9.5.13 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:ACSiso:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:ACSiso:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:ACSiso:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.acsiso.
↪current.fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

return
sflat_curr: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:ACSiso:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.acsiso.
↪current.read(start = 1.0, count = 1.0, decimation = 1.0)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

return
sflat_curr: No help available

6.4.1.9.5.14 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:ACSiso:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:ACSiso:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:ACSiso:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.acsiso.
↪maximum.fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

return
sflat_max: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:ACSiso:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.acsiso.
↪maximum.read(start = 1.0, count = 1.0, decimation = 1.0)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

return
sflat_max: No help available

6.4.1.9.5.15 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:ACSiso:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:ACSiso:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:ACSiso:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.acsiso.
↪minimum.fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

return
sflat_min: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:ACSiso:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.acsiso.
↪minimum.read(start = 1.0, count = 1.0, decimation = 1.0)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

return
sflat_min: No help available

6.4.1.9.5.16 Average

SCPI Command :

```
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

calculate(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:AVERage
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.trace.
↪spectrFlatness.average.calculate(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the spectrum flatness traces for OFDM and OFDMA SISO signals. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
sflat_aver: Comma-separated list power level, one value per subcarrier (including data and pilot subcarriers) The number of subcarriers depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

6.4.1.9.5.17 Current

SCPI Command :

```
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:CURRent
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

calculate(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:CURRent
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.trace.
↪spectrFlatness.current.calculate(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the spectrum flatness traces for OFDM and OFDMA SISO signals. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
sflat_curr: Comma-separated list power level, one value per subcarrier (including data and pilot subcarriers) The number of subcarriers depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

6.4.1.9.5.18 Maximum

SCPI Command :

```
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MAXimum
```

class MaximumCls

Maximum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

calculate(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MAXimum
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.trace.
↪spectrFlatness.maximum.calculate(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the spectrum flatness traces for OFDM and OFDMA SISO signals. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
sflat_max: Comma-separated list power level, one value per subcarrier (including data and pilot subcarriers) The number of subcarriers depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

6.4.1.9.5.19 Mimo

class MimoCls

Mimo commands group definition. 48 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.clone()
```

Subgroups

6.4.1.9.5.20 RxAntenna<RxAntenna>

RepCap Settings

```
# Range: Nr1 .. Nr8
rc = driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.rxAntenna.repcap_rxAntenna_get()
driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.rxAntenna.repcap_rxAntenna_
↪set(repcap.RxAntenna.Nr1)
```

class RxAntennaCls

RxAntenna commands group definition. 48 total commands, 1 Subgroups, 0 group commands Repeated Capability: RxAntenna, default value after init: RxAntenna.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.rxAntenna.clone()
```

Subgroups

6.4.1.9.5.21 Stream<Stream>

RepCap Settings

```
# Range: Nr1 .. Nr8
rc = driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.rxAntenna.stream.repcap_stream_
↪get()
driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.rxAntenna.stream.repcap_stream_
↪set(repcap.Stream.Nr1)
```

class StreamCls

Stream commands group definition. 48 total commands, 6 Subgroups, 0 group commands Repeated Capability: Stream, default value after init: Stream.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.rxAntenna.stream.clone()
```

Subgroups

6.4.1.9.5.22 Acarrier

class AcarrierCls

Acarrier commands group definition. 12 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.rxAntenna.stream.acarrier.
↪clone()
```

Subgroups

6.4.1.9.5.23 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>
↪:ACARrier:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>
↪:ACARrier:AVERage
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam
↪<s>:ACARrier:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:ACARrier:AVERage
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.trace.
↪spectrFlatness.mimo.rxAntenna.stream.acarrier.average.calculate(start = 1.0,␣
↪count = 1.0, decimation = 1.0, rxAntenna = repcap.RxAntenna.Default, stream =␣
↪repcap.Stream.Default)
```

Return the spectrum flatness traces over all carriers (complete FFTSize) for Rx antenna <n> and stream <s>, for true MIMO measurements. The results of the current, average, minimum and maximum traces can be retrieved. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

return
sflat_all_carr_tx: Comma-separated list of power levels, one value per subcarrier (including data, pilot and unused subcarriers) The number of power levels depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:ACARrier:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.
↪rxAntenna.stream.acarrier.average.fetch(start = 1.0, count = 1.0, decimation␣
↪= 1.0, rxAntenna = repcap.RxAntenna.Default, stream = repcap.Stream.Default)
```

Return the spectrum flatness traces over all carriers (complete FFTSize) for Rx antenna <n> and stream <s>, for true MIMO measurements. The results of the current, average, minimum and maximum traces can be retrieved. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

return
sflat_all_carr_tx: Comma-separated list of power levels, one value per subcarrier (including data, pilot and unused subcarriers) The number of power levels depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:ACARrier:AVERage
```

(continues on next page)

(continued from previous page)

```
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.
↪rxAntenna.stream.acarrier.average.read(start = 1.0, count = 1.0, decimation =␣
↪1.0, rxAntenna = repcap.RxAntenna.Default, stream = repcap.Stream.Default)
```

Return the spectrum flatness traces over all carriers (complete FFTSize) for Rx antenna <n> and stream <s>, for true MIMO measurements. The results of the current, average, minimum and maximum traces can be retrieved. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

return
sflat_all_carr_tx: Comma-separated list of power levels, one value per subcarrier (including data, pilot and unused subcarriers) The number of power levels depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

6.4.1.9.5.24 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>
↪:ACARrier:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>
↪:ACARrier:CURRent
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam
↪<s>:ACARrier:CURRent
```

class CurrentCls

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:ACARrier:CURRent
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.trace.
↪spectrFlatness.mimo.rxAntenna.stream.acarrier.current.calculate(start = 1.0,␣
```

(continues on next page)

(continued from previous page)

```
↪count = 1.0, decimation = 1.0, rxAntenna = repcap.RxAntenna.Default, stream =␣
↪repcap.Stream.Default)
```

Return the spectrum flatness traces over all carriers (complete FFTSize) for Rx antenna <n> and stream <s>, for true MIMO measurements. The results of the current, average, minimum and maximum traces can be retrieved. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

return
sflat_all_carr_tx: Comma-separated list of power levels, one value per subcarrier (including data, pilot and unused subcarriers) The number of power levels depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:ACARrier:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.
↪rxAntenna.stream.acarrier.current.fetch(start = 1.0, count = 1.0, decimation␣
↪= 1.0, rxAntenna = repcap.RxAntenna.Default, stream = repcap.Stream.Default)
```

Return the spectrum flatness traces over all carriers (complete FFTSize) for Rx antenna <n> and stream <s>, for true MIMO measurements. The results of the current, average, minimum and maximum traces can be retrieved. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

return
sflat_all_carr_tx: Comma-separated list of power levels, one value per subcarrier (including data, pilot and unused subcarriers) The number of power levels depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:ACARrier:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.
↪rxAntenna.stream.acarrier.current.read(start = 1.0, count = 1.0, decimation =␣
↪1.0, rxAntenna = repcap.RxAntenna.Default, stream = repcap.Stream.Default)
```

Return the spectrum flatness traces over all carriers (complete FFTSize) for Rx antenna <n> and stream <s>, for true MIMO measurements. The results of the current, average, minimum and maximum traces can be retrieved. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

return
sflat_all_carr_tx: Comma-separated list of power levels, one value per subcarrier (including data, pilot and unused subcarriers) The number of power levels depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

6.4.1.9.5.25 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>
↪:ACARrier:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>
↪:ACARrier:MAXimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam
↪<s>:ACARrier:MAXimum
```

class MaximumCls

Maximum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:ACARrier:MAXimum
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.trace.
↪spectrFlatness.mimo.rxAntenna.stream.acarrier.maximum.calculate(start = 1.0,␣
↪count = 1.0, decimation = 1.0, rxAntenna = repcap.RxAntenna.Default, stream =␣
↪repcap.Stream.Default)
```

Return the spectrum flatness traces over all carriers (complete FFTSize) for Rx antenna <n> and stream <s>, for true MIMO measurements. The results of the current, average, minimum and maximum traces can be retrieved. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

return
sflat_all_carr_tx: Comma-separated list of power levels, one value per subcarrier (including data, pilot and unused subcarriers) The number of power levels depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:ACARrier:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.
↪rxAntenna.stream.acarrier.maximum.fetch(start = 1.0, count = 1.0, decimation␣
↪= 1.0, rxAntenna = repcap.RxAntenna.Default, stream = repcap.Stream.Default)
```

Return the spectrum flatness traces over all carriers (complete FFTSize) for Rx antenna <n> and stream <s>, for true MIMO measurements. The results of the current, average, minimum and maximum traces can be retrieved. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

return
sflat_all_carr_tx: Comma-separated list of power levels, one value per subcarrier (including data, pilot and unused subcarriers) The number of power levels depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:ACARrier:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.
↪rxAntenna.stream.acarrier.maximum.read(start = 1.0, count = 1.0, decimation =␣
↪1.0, rxAntenna = repcap.RxAntenna.Default, stream = repcap.Stream.Default)
```

Return the spectrum flatness traces over all carriers (complete FFTSize) for Rx antenna <n> and stream <s>, for true MIMO measurements. The results of the current, average, minimum and maximum traces can be retrieved. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

return
sflat_all_carr_tx: Comma-separated list of power levels, one value per subcarrier (including data, pilot and unused subcarriers) The number of power levels depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

6.4.1.9.5.26 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>
↪:ACARrier:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>
↪:ACARrier:MINimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam
↪<s>:ACARrier:MINimum
```

class MinimumCls

Minimum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:ACARrier:MINimum
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.trace.
↪spectrFlatness.mimo.rxAntenna.stream.acarrier.minimum.calculate(start = 1.0,␣
↪count = 1.0, decimation = 1.0, rxAntenna = repcap.RxAntenna.Default, stream =␣
↪repcap.Stream.Default)
```

Return the spectrum flatness traces over all carriers (complete FFTSize) for Rx antenna <n> and stream <s>, for true MIMO measurements. The results of the current, average, minimum and maximum traces can be retrieved. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

return
sflat_all_carr_tx: Comma-separated list of power levels, one value per subcarrier (including data, pilot and unused subcarriers) The number of power levels depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:ACARrier:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.
↪rxAntenna.stream.acarrier.minimum.fetch(start = 1.0, count = 1.0, decimation␣
↪= 1.0, rxAntenna = repcap.RxAntenna.Default, stream = repcap.Stream.Default)
```

Return the spectrum flatness traces over all carriers (complete FFTSize) for Rx antenna <n> and stream <s>, for true MIMO measurements. The results of the current, average, minimum and maximum traces can be retrieved. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

return
sflat_all_carr_tx: Comma-separated list of power levels, one value per subcarrier (including data, pilot and unused subcarriers) The number of power levels depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:ACARrier:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.
↪rxAntenna.stream.acarrier.minimum.read(start = 1.0, count = 1.0, decimation =␣
↪1.0, rxAntenna = repcap.RxAntenna.Default, stream = repcap.Stream.Default)
```

Return the spectrum flatness traces over all carriers (complete FFTSize) for Rx antenna <n> and stream <s>, for true MIMO measurements. The results of the current, average, minimum and maximum traces

can be retrieved. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

return
sflat_all_carr_tx: Comma-separated list of power levels, one value per subcarrier (including data, pilot and unused subcarriers) The number of power levels depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

6.4.1.9.5.27 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>
↪:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>
↪:AVERage
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam
↪<s>:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:AVERage
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.trace.
↪spectrFlatness.mimo.rxAntenna.stream.average.calculate(start = 1.0, count = 1.
↪0, decimation = 1.0, rxAntenna = repcap.RxAntenna.Default, stream = repcap.
↪Stream.Default)
```

Return the spectrum flatness traces for Rx antenna <n> and stream <s>, for true MIMO measurements. The results of the current, average, minimum and maximum traces can be retrieved. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

return
spec_flat_trace_tx: Comma-separated list power level, one value per subcarrier (including data and pilot subcarriers) The number of subcarriers depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.
↪rxAntenna.stream.average.fetch(start = 1.0, count = 1.0, decimation = 1.0,␣
↪rxAntenna = repcap.RxAntenna.Default, stream = repcap.Stream.Default)
```

Return the spectrum flatness traces for Rx antenna <n> and stream <s>, for true MIMO measurements. The results of the current, average, minimum and maximum traces can be retrieved. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

return
spec_flat_trace_tx: Comma-separated list power level, one value per subcarrier (including data and pilot subcarriers) The number of subcarriers depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.
↪rxAntenna.stream.average.read(start = 1.0, count = 1.0, decimation = 1.0,␣
↪rxAntenna = repcap.RxAntenna.Default, stream = repcap.Stream.Default)
```

Return the spectrum flatness traces for Rx antenna <n> and stream <s>, for true MIMO measurements. The results of the current, average, minimum and maximum traces can be retrieved. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

return
spec_flat_trace_tx: Comma-separated list power level, one value per subcarrier (including data and pilot subcarriers) The number of subcarriers depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

6.4.1.9.5.28 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>
↪:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>
↪:CURRent
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam
↪<s>:CURRent
```

class CurrentCls

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:CURRent
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.trace.
↪spectrFlatness.mimo.rxAntenna.stream.current.calculate(start = 1.0, count = 1.
↪0, decimation = 1.0, rxAntenna = repcap.RxAntenna.Default, stream = repcap.
↪Stream.Default)
```

Return the spectrum flatness traces for Rx antenna <n> and stream <s>, for true MIMO measurements. The results of the current, average, minimum and maximum traces can be retrieved. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

return
spec_flat_trace_tx: Comma-separated list power level, one value per subcarrier (including data and pilot subcarriers) The number of subcarriers depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.
↪rxAntenna.stream.current.fetch(start = 1.0, count = 1.0, decimation = 1.0,␣
↪rxAntenna = repcap.RxAntenna.Default, stream = repcap.Stream.Default)
```

Return the spectrum flatness traces for Rx antenna <n> and stream <s>, for true MIMO measurements. The results of the current, average, minimum and maximum traces can be retrieved. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

return
spec_flat_trace_tx: Comma-separated list power level, one value per subcarrier (including data and pilot subcarriers) The number of subcarriers depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.
↪rxAntenna.stream.current.read(start = 1.0, count = 1.0, decimation = 1.0,␣
↪rxAntenna = repcap.RxAntenna.Default, stream = repcap.Stream.Default)
```

Return the spectrum flatness traces for Rx antenna <n> and stream <s>, for true MIMO measurements. The results of the current, average, minimum and maximum traces can be retrieved. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

return
spec_flat_trace_tx: Comma-separated list power level, one value per subcarrier (including data and pilot subcarriers) The number of subcarriers depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

6.4.1.9.5.29 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>
↪:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>
↪:MAXimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam
↪<s>:MAXimum
```

class MaximumCls

Maximum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:MAXimum
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.trace.
↪spectrFlatness.mimo.rxAntenna.stream.maximum.calculate(start = 1.0, count = 1.
↪0, decimation = 1.0, rxAntenna = repcap.RxAntenna.Default, stream = repcap.
↪Stream.Default)
```

Return the spectrum flatness traces for Rx antenna <n> and stream <s>, for true MIMO measurements. The results of the current, average, minimum and maximum traces can be retrieved. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

return
spec_flat_trace_tx: Comma-separated list power level, one value per subcarrier (including data and pilot subcarriers) The number of subcarriers depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.
↪rxAntenna.stream.maximum.fetch(start = 1.0, count = 1.0, decimation = 1.0,␣
↪rxAntenna = repcap.RxAntenna.Default, stream = repcap.Stream.Default)
```

Return the spectrum flatness traces for Rx antenna <n> and stream <s>, for true MIMO measurements. The results of the current, average, minimum and maximum traces can be retrieved. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

return
spec_flat_trace_tx: Comma-separated list power level, one value per subcarrier (including data and pilot subcarriers) The number of subcarriers depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.
↪rxAntenna.stream.maximum.read(start = 1.0, count = 1.0, decimation = 1.0,␣
↪rxAntenna = repcap.RxAntenna.Default, stream = repcap.Stream.Default)
```

Return the spectrum flatness traces for Rx antenna <n> and stream <s>, for true MIMO measurements. The results of the current, average, minimum and maximum traces can be retrieved. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

return
spec_flat_trace_tx: Comma-separated list power level, one value per subcarrier (including data and pilot subcarriers) The number of subcarriers depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

6.4.1.9.5.30 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>
↪:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>
↪:MINimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam
↪<s>:MINimum
```

class MinimumCls

Minimum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:MINimum
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.trace.
↪spectrFlatness.mimo.rxAntenna.stream.minimum.calculate(start = 1.0, count = 1.
↪0, decimation = 1.0, rxAntenna = repcap.RxAntenna.Default, stream = repcap.
↪Stream.Default)
```

Return the spectrum flatness traces for Rx antenna <n> and stream <s>, for true MIMO measurements. The results of the current, average, minimum and maximum traces can be retrieved. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

return
spec_flat_trace_tx: Comma-separated list power level, one value per subcarrier (including data and pilot subcarriers) The number of subcarriers depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.
↪rxAntenna.stream.minimum.fetch(start = 1.0, count = 1.0, decimation = 1.0,␣
↪rxAntenna = repcap.RxAntenna.Default, stream = repcap.Stream.Default)
```

Return the spectrum flatness traces for Rx antenna <n> and stream <s>, for true MIMO measurements. The results of the current, average, minimum and maximum traces can be retrieved. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

return
spec_flat_trace_tx: Comma-separated list power level, one value per subcarrier (including data and pilot subcarriers) The number of subcarriers depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.
↪rxAntenna.stream.minimum.read(start = 1.0, count = 1.0, decimation = 1.0,␣
↪rxAntenna = repcap.RxAntenna.Default, stream = repcap.Stream.Default)
```

Return the spectrum flatness traces for Rx antenna <n> and stream <s>, for true MIMO measurements. The results of the current, average, minimum and maximum traces can be retrieved. The values described

below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

return
spec_flat_trace_tx: Comma-separated list power level, one value per subcarrier (including data and pilot subcarriers) The number of subcarriers depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

6.4.1.9.5.31 Segment

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.rxAntenna.stream.segment.repcap_
↪segment_get()
driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.rxAntenna.stream.segment.repcap_
↪segment_set(repcap.Segment.Nr1)
```

class SegmentCls

Segment commands group definition. 24 total commands, 5 Subgroups, 0 group commands Repeated Capability: Segment, default value after init: Segment.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.rxAntenna.stream.segment.
↪clone()
```

Subgroups

6.4.1.9.5.32 Acarrier

class AcarrierCls

Acarrier commands group definition. 12 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.rxAntenna.stream.segment.
↪acarrier.clone()
```

Subgroups

6.4.1.9.5.33 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>
↪:SEGMent<seg>:ACARrier:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>
↪:SEGMent<seg>:ACARrier:AVERage
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam
↪<s>:SEGMent<seg>:ACARrier:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*, *segment=Segment.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:SEGMent<seg>
↪:ACARrier:AVERage
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.trace.
↪spectrFlatness.mimo.rxAntenna.stream.segment.acarrier.average.calculate(start␣
↪= 1.0, count = 1.0, decimation = 1.0, rxAntenna = repcap.RxAntenna.Default,␣
↪stream = repcap.Stream.Default, segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spec_flat_trace_segment_tx: No help available

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:SEGMent<seg>
↪:ACARrier:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.
↪rxAntenna.stream.segment.acarrier.average.fetch(start = 1.0, count = 1.0,␣
↪decimation = 1.0, rxAntenna = repcap.RxAntenna.Default, stream = repcap.
↪Stream.Default, segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spec_flat_trace_segment_tx: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:SEGMent<seg>
```

(continues on next page)

(continued from previous page)

```
    ↪:ACARrier:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.
↪rxAntenna.stream.segment.acarrier.average.read(start = 1.0, count = 1.0,␣
↪decimation = 1.0, rxAntenna = repcap.RxAntenna.Default, stream = repcap.
↪Stream.Default, segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spec_flat_trace_segment_tx: No help available

6.4.1.9.5.34 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>
↪:SEGMent<seg>:ACARrier:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>
↪:SEGMent<seg>:ACARrier:CURRent
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam
↪<s>:SEGMent<seg>:ACARrier:CURRent
```

`class CurrentCls`

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

`calculate`(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*, *segment=Segment.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:SEGMent<seg>
↪:ACARrier:CURRent
```

(continues on next page)

(continued from previous page)

```
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.trace.
↪spectrFlatness.mimo.rxAntenna.stream.segment.acarrier.current.calculate(start␣
↪= 1.0, count = 1.0, decimation = 1.0, rxAntenna = repcap.RxAntenna.Default,␣
↪stream = repcap.Stream.Default, segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spec_flat_trace_segment_tx: No help available

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:SEGMent<seg>
↪:ACARrier:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.
↪rxAntenna.stream.segment.acarrier.current.fetch(start = 1.0, count = 1.0,␣
↪decimation = 1.0, rxAntenna = repcap.RxAntenna.Default, stream = repcap.
↪Stream.Default, segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spec_flat_trace_segment_tx: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:SEGMent<seg>
↪:ACARrier:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.
↪rxAntenna.stream.segment.acarrier.current.read(start = 1.0, count = 1.0,␣
↪decimation = 1.0, rxAntenna = repcap.RxAntenna.Default, stream = repcap.
↪Stream.Default, segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spec_flat_trace_segment_tx: No help available

6.4.1.9.5.35 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>
↪:SEGMent<seg>:ACARrier:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>
↪:SEGMent<seg>:ACARrier:MAXimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam
↪<s>:SEGMent<seg>:ACARrier:MAXimum
```

class MaximumCls

Maximum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*, *segment=Segment.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:SEGMent<seg>
↪:ACARrier:MAXimum
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.trace.
↪spectrFlatness.mimo.rxAntenna.stream.segment.acarrier.maximum.calculate(start␣
↪= 1.0, count = 1.0, decimation = 1.0, rxAntenna = repcap.RxAntenna.Default,␣
↪stream = repcap.Stream.Default, segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spec_flat_trace_segment_tx: No help available

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:SEGMent<seg>
↪:ACARrier:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.
↪rxAntenna.stream.segment.acarrier.maximum.fetch(start = 1.0, count = 1.0,␣
↪decimation = 1.0, rxAntenna = repcap.RxAntenna.Default, stream = repcap.
↪Stream.Default, segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spec_flat_trace_segment_tx: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:SEGMent<seg>
↪:ACARrier:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.
↪rxAntenna.stream.segment.acarrier.maximum.read(start = 1.0, count = 1.0,␣
↪decimation = 1.0, rxAntenna = repcap.RxAntenna.Default, stream = repcap.
↪Stream.Default, segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spec_flat_trace_segment_tx: No help available

6.4.1.9.5.36 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>
↪:SEGMent<seg>:ACARrier:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>
↪:SEGMent<seg>:ACARrier:MINimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam
↪<s>:SEGMent<seg>:ACARrier:MINimum
```

class MinimumCls

Minimum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*, *segment=Segment.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:SEGMent<seg>
↪:ACARrier:MINimum
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.trace.
↪spectrFlatness.mimo.rxAntenna.stream.segment.acarrier.minimum.calculate(start␣
↪= 1.0, count = 1.0, decimation = 1.0, rxAntenna = repcap.RxAntenna.Default,␣
↪stream = repcap.Stream.Default, segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spec_flat_trace_segment_tx: No help available

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:SEGMent<seg>
↪:ACARrier:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.
↪rxAntenna.stream.segment.acarrier.minimum.fetch(start = 1.0, count = 1.0,␣
↪decimation = 1.0, rxAntenna = repcap.RxAntenna.Default, stream = repcap.
↪Stream.Default, segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spec_flat_trace_segment_tx: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:SEGMent<seg>
↪:ACARrier:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.
↪rxAntenna.stream.segment.acarrier.minimum.read(start = 1.0, count = 1.0,␣
```

(continues on next page)

(continued from previous page)

```
↪decimation = 1.0, rxAntenna = repcap.RxAntenna.Default, stream = repcap.
↪Stream.Default, segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spec_flat_trace_segment_tx: No help available

6.4.1.9.5.37 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>
↪:SEGMent<seg>:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>
↪:SEGMent<seg>:AVERage
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam
↪<s>:SEGMent<seg>:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*, *segment=Segment.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:SEGMent<seg>:AVERage
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.trace.
↪spectrFlatness.mimo.rxAntenna.stream.segment.average.calculate(start = 1.0,␣
↪count = 1.0, decimation = 1.0, rxAntenna = repcap.RxAntenna.Default, stream =␣
↪repcap.Stream.Default, segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spec_flat_trace_segment_tx: No help available

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:SEGMent<seg>:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.
↪rxAntenna.stream.segment.average.fetch(start = 1.0, count = 1.0, decimation =␣
↪1.0, rxAntenna = repcap.RxAntenna.Default, stream = repcap.Stream.Default,␣
↪segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spec_flat_trace_segment_tx: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:SEGMent<seg>:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.
↪rxAntenna.stream.segment.average.read(start = 1.0, count = 1.0, decimation =␣
↪1.0, rxAntenna = repcap.RxAntenna.Default, stream = repcap.Stream.Default,␣
↪segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spec_flat_trace_segment_tx: No help available

6.4.1.9.5.38 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>
↪:SEGMent<seg>:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>
↪:SEGMent<seg>:CURRent
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam
↪<s>:SEGMent<seg>:CURRent
```

class CurrentCls

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*, *segment=Segment.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:SEGMent<seg>:CURRent
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.trace.
↪spectrFlatness.mimo.rxAntenna.stream.segment.current.calculate(start = 1.0,␣
↪count = 1.0, decimation = 1.0, rxAntenna = repcap.RxAntenna.Default, stream =␣
↪repcap.Stream.Default, segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spec_flat_trace_segment_tx: No help available

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:SEGMent<seg>:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.
↪rxAntenna.stream.segment.current.fetch(start = 1.0, count = 1.0, decimation =␣
↪1.0, rxAntenna = repcap.RxAntenna.Default, stream = repcap.Stream.Default,␣
↪segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spec_flat_trace_segment_tx: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:SEGMent<seg>:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.
↪rxAntenna.stream.segment.current.read(start = 1.0, count = 1.0, decimation =␣
↪1.0, rxAntenna = repcap.RxAntenna.Default, stream = repcap.Stream.Default,␣
↪segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spec_flat_trace_segment_tx: No help available

6.4.1.9.5.39 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>
↪:SEGMent<seg>:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>
↪:SEGMent<seg>:MAXimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam
↪<s>:SEGMent<seg>:MAXimum
```

class MaximumCls

Maximum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*, *segment=Segment.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:SEGMent<seg>:MAXimum
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.trace.
↪spectrFlatness.mimo.rxAntenna.stream.segment.maximum.calculate(start = 1.0,␣
↪count = 1.0, decimation = 1.0, rxAntenna = repcap.RxAntenna.Default, stream =␣
↪repcap.Stream.Default, segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spec_flat_trace_segment_tx: No help available

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:SEGMent<seg>:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.
↪rxAntenna.stream.segment.maximum.fetch(start = 1.0, count = 1.0, decimation =␣
↪1.0, rxAntenna = repcap.RxAntenna.Default, stream = repcap.Stream.Default,␣
↪segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spec_flat_trace_segment_tx: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:SEGMent<seg>:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.
↪rxAntenna.stream.segment.maximum.read(start = 1.0, count = 1.0, decimation =␣
↪1.0, rxAntenna = repcap.RxAntenna.Default, stream = repcap.Stream.Default,␣
↪segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spec_flat_trace_segment_tx: No help available

6.4.1.9.5.40 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>
↪:SEGMent<seg>:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>
↪:SEGMent<seg>:MINimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam
↪<s>:SEGMent<seg>:MINimum
```

class MinimumCls

Minimum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*, *segment=Segment.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:SEGMent<seg>:MINimum
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.trace.
↪spectrFlatness.mimo.rxAntenna.stream.segment.minimum.calculate(start = 1.0,␣
↪count = 1.0, decimation = 1.0, rxAntenna = repcap.RxAntenna.Default, stream =␣
↪repcap.Stream.Default, segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spec_flat_trace_segment_tx: No help available

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:SEGMent<seg>:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.
↪rxAntenna.stream.segment.minimum.fetch(start = 1.0, count = 1.0, decimation =␣
↪1.0, rxAntenna = repcap.RxAntenna.Default, stream = repcap.Stream.Default,␣
↪segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spec_flat_trace_segment_tx: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *rxAntenna=RxAntenna.Default*, *stream=Stream.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:STReam<s>:SEGMent<seg>:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.mimo.
↪rxAntenna.stream.segment.minimum.read(start = 1.0, count = 1.0, decimation =␣
↪1.0, rxAntenna = repcap.RxAntenna.Default, stream = repcap.Stream.Default,␣
↪segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param rxAntenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'RxAntenna')

param stream
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stream')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spec_flat_trace_segment_tx: No help available

6.4.1.9.5.41 Minimum

SCPI Command :

```
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MINimum
```

class MinimumCls

Minimum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

calculate(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MINimum
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.trace.
↪spectrFlatness.minimum.calculate(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the spectrum flatness traces for OFDM and OFDMA SISO signals. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
sflat_min: Comma-separated list power level, one value per subcarrier (including data

and pilot subcarriers) The number of subcarriers depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

6.4.1.9.5.42 Ofdm

class OfdmCls

Ofdm commands group definition. 8 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.spectrFlatness.ofdm.clone()
```

Subgroups

6.4.1.9.5.43 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness[:OFDM]:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness[:OFDM]:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness[:OFDM]:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.ofdm.
↪average.fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the spectrum flatness traces for OFDM and OFDMA SISO signals. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
sflat_aver: Comma-separated list power level, one value per subcarrier (including data and pilot subcarriers) The number of subcarriers depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness[:OFDM]:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.ofdm.
↪average.read(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the spectrum flatness traces for OFDM and OFDMA SISO signals. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
sflat_aver: Comma-separated list power level, one value per subcarrier (including data and pilot subcarriers) The number of subcarriers depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

6.4.1.9.5.44 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness[:OFDM]:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness[:OFDM]:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness[:OFDM]:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.ofdm.
↪current.fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the spectrum flatness traces for OFDM and OFDMA SISO signals. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
sflat_curr: Comma-separated list power level, one value per subcarrier (including data and pilot subcarriers) The number of subcarriers depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness[:OFDM]:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.ofdm.
↪current.read(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the spectrum flatness traces for OFDM and OFDMA SISO signals. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
sflat_curr: Comma-separated list power level, one value per subcarrier (including data and pilot subcarriers) The number of subcarriers depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

6.4.1.9.5.45 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness[:OFDM]:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness[:OFDM]:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness[:OFDM]:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.ofdm.
↪maximum.fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the spectrum flatness traces for OFDM and OFDMA SISO signals. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
sflat_max: Comma-separated list power level, one value per subcarrier (including data and pilot subcarriers) The number of subcarriers depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness[:OFDM]:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.ofdm.
↪maximum.read(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the spectrum flatness traces for OFDM and OFDMA SISO signals. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
sflat_max: Comma-separated list power level, one value per subcarrier (including data and pilot subcarriers) The number of subcarriers depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

6.4.1.9.5.46 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness[:OFDM]:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness[:OFDM]:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness[:OFDM]:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.ofdm.
↪minimum.fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the spectrum flatness traces for OFDM and OFDMA SISO signals. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
sflat_min: Comma-separated list power level, one value per subcarrier (including data and pilot subcarriers) The number of subcarriers depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TRACe:SFLatness[:OFDM]:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.ofdm.
↪minimum.read(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the spectrum flatness traces for OFDM and OFDMA SISO signals. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
sflat_min: Comma-separated list power level, one value per subcarrier (including data and pilot subcarriers) The number of subcarriers depends on the WLAN standard, channel bandwidth and mode, see Table 'OFDM subcarriers'.

6.4.1.9.5.47 Segment

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.wlanMeas.multiEval.trace.spectrFlatness.segment.repcap_segment_get()
driver.wlanMeas.multiEval.trace.spectrFlatness.segment.repcap_segment_set(repcap.Segment.
↪Nr1)
```

class SegmentCls

Segment commands group definition. 12 total commands, 4 Subgroups, 0 group commands Repeated Capability: Segment, default value after init: Segment.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.spectrFlatness.segment.clone()
```

Subgroups

6.4.1.9.5.48 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SEGMent<seg>:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SEGMent<seg>:AVERage
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SEGMent<seg>:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SEGMent
↪<seg>:AVERage
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.trace.
↪spectrFlatness.segment.average.calculate(start = 1.0, count = 1.0, decimation␣
↪= 1.0, segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spec_flat_trace_segment_aver: No help available

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SEGMent
↪<seg>:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.segment.
↪average.fetch(start = 1.0, count = 1.0, decimation = 1.0, segment = repcap.
↪Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spec_flat_trace_segment_aver: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SEGMent<seg>
↪:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.segment.
↪average.read(start = 1.0, count = 1.0, decimation = 1.0, segment = repcap.
↪Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spec_flat_trace_segment_aver: No help available

6.4.1.9.5.49 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SEGMent<seg>:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SEGMent<seg>:CURRent
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SEGMent<seg>:CURRent
```

class CurrentCls

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SEGMent
↪<seg>:CURRent
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.trace.
↪spectrFlatness.segment.current.calculate(start = 1.0, count = 1.0, decimation␣
↪= 1.0, segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spec_flat_trace_segment_curr: No help available

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SEGMent
↪<seg>:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.segment.
↪current.fetch(start = 1.0, count = 1.0, decimation = 1.0, segment = repcap.
↪Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spec_flat_trace_segment_curr: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SEGMent<seg>
↪:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.segment.
↪current.read(start = 1.0, count = 1.0, decimation = 1.0, segment = repcap.
↪Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spec_flat_trace_segment_curr: No help available

6.4.1.9.5.50 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SEGMent<seg>:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SEGMent<seg>:MAXimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SEGMent<seg>:MAXimum
```

class MaximumCls

Maximum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SEGMent
↪<seg>:MAXimum
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.trace.
↪spectrFlatness.segment.maximum.calculate(start = 1.0, count = 1.0, decimation␣
↪= 1.0, segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spec_flat_trace_segment_max: No help available

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SEGMent
↪<seg>:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.segment.
↪maximum.fetch(start = 1.0, count = 1.0, decimation = 1.0, segment = repcap.
↪Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spec_flat_trace_segment_max: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SEGMent<seg>
↪:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.segment.
↪maximum.read(start = 1.0, count = 1.0, decimation = 1.0, segment = repcap.
↪Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spec_flat_trace_segment_max: No help available

6.4.1.9.5.51 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SEGMent<seg>:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SEGMent<seg>:MINimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SEGMent<seg>:MINimum
```

class MinimumCls

Minimum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SEGMent
↪<seg>:MINimum
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.trace.
↪spectrFlatness.segment.minimum.calculate(start = 1.0, count = 1.0, decimation␣
↪= 1.0, segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spec_flat_trace_segment_min: No help available

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SEGMent
↪<seg>:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.segment.
```

(continues on next page)

(continued from previous page)

```
↪minimum.fetch(start = 1.0, count = 1.0, decimation = 1.0, segment = repcap.
↪Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spec_flat_trace_segment_min: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SEGMent<seg>
↪:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.spectrFlatness.segment.
↪minimum.read(start = 1.0, count = 1.0, decimation = 1.0, segment = repcap.
↪Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spec_flat_trace_segment_min: No help available

6.4.1.9.6 Terror

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TERRor
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TERRor
```

class TerrorCls

Terror commands group definition. 4 total commands, 1 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TERRor
value: List[float] = driver.wlanMeas.multiEval.trace.terror.fetch(start = 1.0,␣
↪count = 1.0, decimation = 1.0)
```

Return timing error traces. The number of results corresponds to the statistic count, see method RsCMPX_WlanMeas. Configure.WlanMeas.MultiEval.Scount.powerVsTime.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
te_times: Comma-separated list of timing error values

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TERRor
value: List[float] = driver.wlanMeas.multiEval.trace.terror.read(start = 1.0,␣
↪count = 1.0, decimation = 1.0)
```

Return timing error traces. The number of results corresponds to the statistic count, see method RsCMPX_WlanMeas. Configure.WlanMeas.MultiEval.Scount.powerVsTime.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
te_times: Comma-separated list of timing error values

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.terror.clone()
```

Subgroups

6.4.1.9.6.1 Mimo<Mimo>

RepCap Settings

```
# Range: Nr1 .. Nr8
rc = driver.wlanMeas.multiEval.trace.terror.mimo.repcap_mimo_get()
driver.wlanMeas.multiEval.trace.terror.mimo.repcap_mimo_set(repcap.Mimo.Nr1)
```

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TERRor:MIMO<n>
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TERRor:MIMO<n>
```

class MimoCls

Mimo commands group definition. 2 total commands, 0 Subgroups, 2 group commands Repeated Capability: Mimo, default value after init: Mimo.Nr1

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TERRor:MIMO<n>
value: List[float] = driver.wlanMeas.multiEval.trace.terror.mimo.fetch(start =␣
↪1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.Default)
```

Return timing error traces for MIMO. The number of results corresponds to the statistic count, see method RsCMPX_WlanMeas. Configure.WlanMeas.MultiEval.Scount.powerVsTime.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
te_times: Comma-separated list of timing error values

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TERRor:MIMO<n>
value: List[float] = driver.wlanMeas.multiEval.trace.terror.mimo.read(start = 1.
↪0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.Default)
```

Return timing error traces for MIMO. The number of results corresponds to the statistic count, see method RsCMPX_WlanMeas. Configure.WlanMeas.MultiEval.Scount.powerVsTime.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
te_times: Comma-separated list of timing error values

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.terror.mimo.clone()
```

6.4.1.9.7 TsMask

class TsMaskCls

TsMask commands group definition. 42 total commands, 8 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.tsMask.clone()
```

Subgroups

6.4.1.9.7.1 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.tsMask.average.fetch(start␣
↪= 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the transmit spectrum mask traces for SISO measurements and bandwidths with one segment. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
spectrum_trace: Comma-separated list of power values, trace from left to right

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.tsMask.average.read(start␣
↪= 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the transmit spectrum mask traces for SISO measurements and bandwidths with one segment. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
spectrum_trace: Comma-separated list of power values, trace from left to right

6.4.1.9.7.2 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.tsMask.current.fetch(start␣
↪= 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the transmit spectrum mask traces for SISO measurements and bandwidths with one segment. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
spectrum_trace: Comma-separated list of power values, trace from left to right

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.tsMask.current.read(start␣
↪= 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the transmit spectrum mask traces for SISO measurements and bandwidths with one segment. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
spectrum_trace: Comma-separated list of power values, trace from left to right

6.4.1.9.7.3 Frequency

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:FREQuency
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:FREQuency
```

class FrequencyCls

Frequency commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:FREQuency
value: List[float] = driver.wlanMeas.multiEval.trace.tsMask.frequency.
↪fetch(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the frequency values (X-values) of the transmit spectrum mask limit line trace, for SISO and bandwidths with one or two segments.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
spec_trace_freq: Comma-separated list of values, trace from left to right 0 Hz corresponds to the center of the channel.

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:FREQuency
value: List[float] = driver.wlanMeas.multiEval.trace.tsMask.frequency.
↪read(start = 1.0, count = 1.0, decimation = 1.0)
```

Return the frequency values (X-values) of the transmit spectrum mask limit line trace, for SISO and bandwidths with one or two segments.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
spec_trace_freq: Comma-separated list of values, trace from left to right 0 Hz corresponds to the center of the channel.

6.4.1.9.7.4 Mask

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MASK
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MASK
```

class MaskCls

Mask commands group definition. 8 total commands, 2 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MASK
value: List[float] = driver.wlanMeas.multiEval.trace.tsMask.mask.fetch(start =␣
↪1.0, count = 1.0, decimation = 1.0)
```

Return the power values (Y-values) of the transmit spectrum mask limit line trace, for SISO measurements and bandwidths with one segment.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
spec_trace_mask: Comma-separated list of power values, trace from left to right

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MASK
value: List[float] = driver.wlanMeas.multiEval.trace.tsMask.mask.read(start = 1.
↪0, count = 1.0, decimation = 1.0)
```

Return the power values (Y-values) of the transmit spectrum mask limit line trace, for SISO measurements and bandwidths with one segment.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
spec_trace_mask: Comma-separated list of power values, trace from left to right

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.tsMask.mask.clone()
```

Subgroups

6.4.1.9.7.5 Mimo<Mimo>

RepCap Settings

```
# Range: Nr1 .. Nr8
rc = driver.wlanMeas.multiEval.trace.tsMask.mask.mimo.repcap_mimo_get()
driver.wlanMeas.multiEval.trace.tsMask.mask.mimo.repcap_mimo_set(repcap.Mimo.Nr1)
```

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MASK:MIMO<n>
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MASK:MIMO<n>
```

class MimoCls

Mimo commands group definition. 4 total commands, 1 Subgroups, 2 group commands Repeated Capability: Mimo, default value after init: Mimo.Nr1

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MASK:MIMO<n>
value: List[float] = driver.wlanMeas.multiEval.trace.tsMask.mask.mimo.
↪fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.Default)
```

Return the power values (Y-values) of the transmit spectrum mask limit line trace, for MIMO measurements, antenna <n>, bandwidths with one segment.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
spec_trace_mask_tx: Comma-separated list of power values, trace from left to right

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MASK:MIMO<n>
value: List[float] = driver.wlanMeas.multiEval.trace.tsMask.mask.mimo.
↪read(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.Default)
```

Return the power values (Y-values) of the transmit spectrum mask limit line trace, for MIMO measurements, antenna <n>, bandwidths with one segment.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
spec_trace_mask_tx: Comma-separated list of power values, trace from left to right

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.tsMask.mask.mimo.clone()
```

Subgroups

6.4.1.9.7.6 Segment

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.wlanMeas.multiEval.trace.tsMask.mask.mimo.segment.repcap_segment_get()
driver.wlanMeas.multiEval.trace.tsMask.mask.mimo.segment.repcap_segment_set(repcap.
↪Segment.Nr1)
```

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MASK:MIMO<n>:SEGMent<seg>
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MASK:MIMO<n>:SEGMent<seg>
```

class SegmentCls

Segment commands group definition. 2 total commands, 0 Subgroups, 2 group commands Repeated Capability: Segment, default value after init: Segment.Nr1

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MASK:MIMO<n>
↪:SEGMent<seg>
value: List[float] = driver.wlanMeas.multiEval.trace.tsMask.mask.mimo.segment.
↪fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.Default,␣
↪segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spec_trace_mask_tx: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MASK:MIMO<n>
↪:SEGMent<seg>
value: List[float] = driver.wlanMeas.multiEval.trace.tsMask.mask.mimo.segment.
↪read(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.Default,␣
↪segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spec_trace_mask_tx: No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.tsMask.mask.mimo.segment.clone()
```

6.4.1.9.7.7 Segment

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.wlanMeas.multiEval.trace.tsMask.mask.segment.repcap_segment_get()
driver.wlanMeas.multiEval.trace.tsMask.mask.segment.repcap_segment_set(repcap.Segment.
↪Nr1)
```

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MASK:SEGMent<seg>
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MASK:SEGMent<seg>
```

class SegmentCls

Segment commands group definition. 2 total commands, 0 Subgroups, 2 group commands Repeated Capability: Segment, default value after init: Segment.Nr1

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MASK:SEGMent
↪<seg>
value: List[float] = driver.wlanMeas.multiEval.trace.tsMask.mask.segment.
↪fetch(start = 1.0, count = 1.0, decimation = 1.0, segment = repcap.Segment.
↪Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spec_trace_mask: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MASK:SEGMent
↪<seg>
value: List[float] = driver.wlanMeas.multiEval.trace.tsMask.mask.segment.
↪read(start = 1.0, count = 1.0, decimation = 1.0, segment = repcap.Segment.
↪Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spec_trace_mask: No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.tsMask.mask.segment.clone()
```

6.4.1.9.7.8 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.tsMask.maximum.fetch(start␣
↪= 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the transmit spectrum mask traces for SISO measurements and bandwidths with one segment. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
spectrum_trace: Comma-separated list of power values, trace from left to right

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.tsMask.maximum.read(start
↪= 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the transmit spectrum mask traces for SISO measurements and bandwidths with one segment. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
spectrum_trace: Comma-separated list of power values, trace from left to right

6.4.1.9.7.9 Mimo<Mimo>

RepCap Settings

```
# Range: Nr1 .. Nr8
rc = driver.wlanMeas.multiEval.trace.tsMask.mimo.repcap_mimo_get()
driver.wlanMeas.multiEval.trace.tsMask.mimo.repcap_mimo_set(repcap.Mimo.Nr1)
```

class MimoCls

Mimo commands group definition. 16 total commands, 5 Subgroups, 0 group commands Repeated Capability: Mimo, default value after init: Mimo.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.tsMask.mimo.clone()
```

Subgroups

6.4.1.9.7.10 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>
↪:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.tsMask.mimo.average.
↪fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.Default)
```

Return the values of the transmit spectrum mask traces for MIMO measurements, antenna <n>, bandwidths with one segment. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
spectrum_trace_tx: Comma-separated list of power values, trace from left to right

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.tsMask.mimo.average.
↪read(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.Default)
```

Return the values of the transmit spectrum mask traces for MIMO measurements, antenna <n>, bandwidths with one segment. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
spectrum_trace_tx: Comma-separated list of power values, trace from left to right

6.4.1.9.7.11 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>
↪:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.tsMask.mimo.current.
↪fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.Default)
```

Return the values of the transmit spectrum mask traces for MIMO measurements, antenna <n>, bandwidths with one segment. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
spectrum_trace_tx: Comma-separated list of power values, trace from left to right

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.tsMask.mimo.current.
↪read(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.Default)
```

Return the values of the transmit spectrum mask traces for MIMO measurements, antenna <n>, bandwidths with one segment. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
spectrum_trace_tx: Comma-separated list of power values, trace from left to right

6.4.1.9.7.12 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>
↪:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.tsMask.mimo.maximum.
↪fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.Default)
```

Return the values of the transmit spectrum mask traces for MIMO measurements, antenna <n>, bandwidths with one segment. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
spectrum_trace_tx: Comma-separated list of power values, trace from left to right

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.tsMask.mimo.maximum.
↪read(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.Default)
```

Return the values of the transmit spectrum mask traces for MIMO measurements, antenna <n>, bandwidths with one segment. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
spectrum_trace_tx: Comma-separated list of power values, trace from left to right

6.4.1.9.7.13 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>
↪:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.tsMask.mimo.minimum.
↪fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.Default)
```

Return the values of the transmit spectrum mask traces for MIMO measurements, antenna <n>, bandwidths with one segment. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
spectrum_trace_tx: Comma-separated list of power values, trace from left to right

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.tsMask.mimo.minimum.
↪read(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.Default)
```

Return the values of the transmit spectrum mask traces for MIMO measurements, antenna <n>, bandwidths with one segment. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
spectrum_trace_tx: Comma-separated list of power values, trace from left to right

6.4.1.9.7.14 Segment

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.wlanMeas.multiEval.trace.tsMask.mimo.segment.repcap_segment_get()
driver.wlanMeas.multiEval.trace.tsMask.mimo.segment.repcap_segment_set(repcap.Segment.
↪Nr1)
```

class SegmentCls

Segment commands group definition. 8 total commands, 4 Subgroups, 0 group commands Repeated Capability: Segment, default value after init: Segment.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.tsMask.mimo.segment.clone()
```

Subgroups

6.4.1.9.7.15 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>:SEGMent<seg>:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>:SEGMent<seg>:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>
↪:SEGMent<seg>:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.tsMask.mimo.segment.
↪average.fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.
↪Default, segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spectrum_trace_tx: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>:SEGMent
↪<seg>:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.tsMask.mimo.segment.
↪average.read(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.
↪Default, segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spectrum_trace_tx: No help available

6.4.1.9.7.16 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>:SEGMent<seg>:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>:SEGMent<seg>:CURRent
```

`class CurrentCls`

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

`fetch`(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>
↪:SEGMent<seg>:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.tsMask.mimo.segment.
↪current.fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.
↪Default, segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spectrum_trace_tx: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>:SEGMent
↪<seg>:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.tsMask.mimo.segment.
↪current.read(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.
↪Default, segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spectrum_trace_tx: No help available

6.4.1.9.7.17 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>:SEGMent<seg>:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>:SEGMent<seg>:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>
↪:SEGMent<seg>:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.tsMask.mimo.segment.
↪maximum.fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.
↪Default, segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spectrum_trace_tx: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>:SEGMent
↪<seg>:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.tsMask.mimo.segment.
↪maximum.read(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.
↪Default, segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spectrum_trace_tx: No help available

6.4.1.9.7.18 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>:SEGMent<seg>:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>:SEGMent<seg>:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>
↪:SEGMent<seg>:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.tsMask.mimo.segment.
↪minimum.fetch(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.
↪Default, segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spectrum_trace_tx: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *mimo=Mimo.Default*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>:SEGMent
↪<seg>:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.tsMask.mimo.segment.
↪minimum.read(start = 1.0, count = 1.0, decimation = 1.0, mimo = repcap.Mimo.
↪Default, segment = repcap.Segment.Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spectrum_trace_tx: No help available

6.4.1.9.7.19 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.tsMask.minimum.fetch(start␣
↪= 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the transmit spectrum mask traces for SISO measurements and bandwidths with one segment. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
spectrum_trace: Comma-separated list of power values, trace from left to right

read(*start: float = None*, *count: float = None*, *decimation: float = None*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.tsMask.minimum.read(start
↪= 1.0, count = 1.0, decimation = 1.0)
```

Return the values of the transmit spectrum mask traces for SISO measurements and bandwidths with one segment. The results of the current, average, minimum and maximum traces can be retrieved.

Suppressed linked return values: reliability

param start
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param count
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

param decimation
For the optional query parameters start, count and decimation, see 'Trace subarrays'.

return
spectrum_trace: Comma-separated list of power values, trace from left to right

6.4.1.9.7.20 Segment

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.wlanMeas.multiEval.trace.tsMask.segment.repcap_segment_get()
driver.wlanMeas.multiEval.trace.tsMask.segment.repcap_segment_set(repcap.Segment.Nr1)
```

class SegmentCls

Segment commands group definition. 8 total commands, 4 Subgroups, 0 group commands Repeated Capability: Segment, default value after init: Segment.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.trace.tsMask.segment.clone()
```

Subgroups

6.4.1.9.7.21 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:SEGMent<seg>:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:SEGMent<seg>:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:SEGMent<seg>
↪:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.tsMask.segment.average.
↪fetch(start = 1.0, count = 1.0, decimation = 1.0, segment = repcap.Segment.
↪Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spectrum_trace_segment: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:SEGMent<seg>
↪:AVERage
value: List[float] = driver.wlanMeas.multiEval.trace.tsMask.segment.average.
↪read(start = 1.0, count = 1.0, decimation = 1.0, segment = repcap.Segment.
↪Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spectrum_trace_segment: No help available

6.4.1.9.7.22 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:SEGMent<seg>:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:SEGMent<seg>:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:SEGMent<seg>
↪:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.tsMask.segment.current.
↪fetch(start = 1.0, count = 1.0, decimation = 1.0, segment = repcap.Segment.
↪Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spectrum_trace_segment: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:SEGMent<seg>
↪:CURRent
value: List[float] = driver.wlanMeas.multiEval.trace.tsMask.segment.current.
↪read(start = 1.0, count = 1.0, decimation = 1.0, segment = repcap.Segment.
↪Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spectrum_trace_segment: No help available

6.4.1.9.7.23 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:SEGMent<seg>:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:SEGMent<seg>:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:SEGMent<seg>
↪:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.tsMask.segment.maximum.
↪fetch(start = 1.0, count = 1.0, decimation = 1.0, segment = repcap.Segment.
↪Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spectrum_trace_segment: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:SEGMent<seg>
↪:MAXimum
value: List[float] = driver.wlanMeas.multiEval.trace.tsMask.segment.maximum.
↪read(start = 1.0, count = 1.0, decimation = 1.0, segment = repcap.Segment.
↪Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spectrum_trace_segment: No help available

6.4.1.9.7.24 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:SEGMent<seg>:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:SEGMent<seg>:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:SEGMent<seg>
↪:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.tsMask.segment.minimum.
↪fetch(start = 1.0, count = 1.0, decimation = 1.0, segment = repcap.Segment.
↪Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spectrum_trace_segment: No help available

read(*start: float = None*, *count: float = None*, *decimation: float = None*, *segment=Segment.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:SEGMent<seg>
↪:MINimum
value: List[float] = driver.wlanMeas.multiEval.trace.tsMask.segment.minimum.
↪read(start = 1.0, count = 1.0, decimation = 1.0, segment = repcap.Segment.
↪Default)
```

No command help available

Suppressed linked return values: reliability

param start
No help available

param count
No help available

param decimation
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
spectrum_trace_segment: No help available

6.4.1.10 TsMask

class TsMaskCls

TsMask commands group definition. 140 total commands, 12 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.tsMask.clone()
```

Subgroups

6.4.1.10.1 Acsiso

class AcsisoCls

Acsiso commands group definition. 9 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.tsMask.acsiso.clone()
```

Subgroups

6.4.1.10.1.1 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:ACSiso:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:ACSiso:AVERage
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:ACSiso:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Ab_Aver: enums.ResultStatus2: No parameter help available
- Bc_Aver: enums.ResultStatus2: No parameter help available
- Cd_Aver: enums.ResultStatus2: No parameter help available
- De_Aver: enums.ResultStatus2: No parameter help available
- Ed_Aver: enums.ResultStatus2: No parameter help available
- Dc_Aver: enums.ResultStatus2: No parameter help available
- Cb_Aver: enums.ResultStatus2: No parameter help available
- Ba_Aver: enums.ResultStatus2: No parameter help available
- Out_Of_Tol: enums.ResultStatus2: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Ab_Aver: float: No parameter help available
- Bc_Aver: float: No parameter help available
- Cd_Aver: float: No parameter help available
- De_Aver: float: No parameter help available
- Ed_Aver: float: No parameter help available
- Dc_Aver: float: No parameter help available
- Cb_Aver: float: No parameter help available
- Ba_Aver: float: No parameter help available
- Out_Of_Tol: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:ACSiso:AVERage
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.acsiso.average.
↪calculate()
```

No command help available

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:ACSiso:AVERage
value: ResultData = driver.wlanMeas.multiEval.tsMask.acsiso.average.fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:ACSiso:AVERage
value: ResultData = driver.wlanMeas.multiEval.tsMask.acsiso.average.read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.10.1.2 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:ACSiso:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:ACSiso:CURRent
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:ACSiso:CURRent
```

class CurrentCls

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Ab_Curr: enums.ResultStatus2: No parameter help available
- Bc_Curr: enums.ResultStatus2: No parameter help available
- Cd_Curr: enums.ResultStatus2: No parameter help available
- De_Curr: enums.ResultStatus2: No parameter help available
- Ed_Curr: enums.ResultStatus2: No parameter help available
- Dc_Curr: enums.ResultStatus2: No parameter help available
- Cb_Curr: enums.ResultStatus2: No parameter help available
- Ba_Curr: enums.ResultStatus2: No parameter help available
- Out_Of_Tol: enums.ResultStatus2: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Ab_Curr: float: No parameter help available
- Bc_Curr: float: No parameter help available
- Cd_Curr: float: No parameter help available
- De_Curr: float: No parameter help available
- Ed_Curr: float: No parameter help available
- Dc_Curr: float: No parameter help available
- Cb_Curr: float: No parameter help available
- Ba_Curr: float: No parameter help available
- Out_Of_Tol: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:ACSiso:CURRent
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.acsiso.current.
↪calculate()
```

No command help available

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:ACSiso:CURRent
value: ResultData = driver.wlanMeas.multiEval.tsMask.acsiso.current.fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:ACSiso:CURRent
value: ResultData = driver.wlanMeas.multiEval.tsMask.acsiso.current.read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.10.1.3 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:ACSiso:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:ACSiso:MAXimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:ACSiso:MAXimum
```

class MaximumCls

Maximum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Ab_Max: enums.ResultStatus2: No parameter help available
- Bc_Max: enums.ResultStatus2: No parameter help available
- Cd_Max: enums.ResultStatus2: No parameter help available
- De_Max: enums.ResultStatus2: No parameter help available
- Ed_Max: enums.ResultStatus2: No parameter help available
- Dcmax: enums.ResultStatus2: No parameter help available
- Cb_Max: enums.ResultStatus2: No parameter help available
- Ba_Max: enums.ResultStatus2: No parameter help available
- Out_Of_Tol: enums.ResultStatus2: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Ab_Max: float: No parameter help available
- Bc_Max: float: No parameter help available
- Cd_Max: float: No parameter help available
- De_Max: float: No parameter help available
- Ed_Max: float: No parameter help available
- Dcmax: float: No parameter help available
- Cb_Max: float: No parameter help available
- Ba_Max: float: No parameter help available
- Out_Of_Tol: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:ACSiso:MAXimum
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.acsiso.maximum.
↪calculate()
```

No command help available

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:ACSiso:MAXimum
value: ResultData = driver.wlanMeas.multiEval.tsMask.acsiso.maximum.fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:ACSiso:MAXimum
value: ResultData = driver.wlanMeas.multiEval.tsMask.acsiso.maximum.read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.10.2 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:AVERage
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tol: enums.ResultStatus2: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified transmit spectrum mask limits.
- Margin: List[enums.ResultStatus2]: Comma-separated list of margin values, one value per spectrum mask area The number of margin values depends on the selected standard, see Table 'Spectrum mask areas'.

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified transmit spectrum mask limits.
- Margin: List[float]: Comma-separated list of margin values, one value per spectrum mask area The number of margin values depends on the selected standard, see Table 'Spectrum mask areas'.

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:AVERage
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.average.calculate()
```

Return the limit line margin values of the transmit spectrum mask for SISO measurements and bandwidths with one segment. Margins for the current, average, minimum and maximum traces are returned. A positive result indicates that the trace is located above the limit line. The limit is exceeded. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:AVERage
value: ResultData = driver.wlanMeas.multiEval.tsMask.average.fetch()
```

Return the limit line margin values of the transmit spectrum mask for SISO measurements and bandwidths with one segment. Margins for the current, average, minimum and maximum traces are returned. A positive result indicates that the trace is located above the limit line. The limit is exceeded. The values described

below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:AVERage
value: ResultData = driver.wlanMeas.multiEval.tsMask.average.read()
```

Return the limit line margin values of the transmit spectrum mask for SISO measurements and bandwidths with one segment. Margins for the current, average, minimum and maximum traces are returned. A positive result indicates that the trace is located above the limit line. The limit is exceeded. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.10.3 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:CURRent
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:CURRent
```

class CurrentCls

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tol: enums.ResultStatus2: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified transmit spectrum mask limits.
- Margin: List[enums.ResultStatus2]: Comma-separated list of margin values, one value per spectrum mask area The number of margin values depends on the selected standard, see Table 'Spectrum mask areas'.

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified transmit spectrum mask limits.
- Margin: List[float]: Comma-separated list of margin values, one value per spectrum mask area The number of margin values depends on the selected standard, see Table 'Spectrum mask areas'.

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:CURRent
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.current.calculate()
```

Return the limit line margin values of the transmit spectrum mask for SISO measurements and bandwidths with one segment. Margins for the current, average, minimum and maximum traces are returned. A positive result indicates that the trace is located above the limit line. The limit is exceeded. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:CURRent
value: ResultData = driver.wlanMeas.multiEval.tsMask.current.fetch()
```

Return the limit line margin values of the transmit spectrum mask for SISO measurements and bandwidths with one segment. Margins for the current, average, minimum and maximum traces are returned. A positive result indicates that the trace is located above the limit line. The limit is exceeded. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:CURRent
value: ResultData = driver.wlanMeas.multiEval.tsMask.current.read()
```

Return the limit line margin values of the transmit spectrum mask for SISO measurements and bandwidths with one segment. Margins for the current, average, minimum and maximum traces are returned. A positive result indicates that the trace is located above the limit line. The limit is exceeded. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.10.4 Dsss

class DsssCls

Dsss commands group definition. 9 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.tsMask.dsss.clone()
```

Subgroups

6.4.1.10.4.1 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:DSSS:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:DSSS:AVERage
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:DSSS:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Ab_Aver: enums.ResultStatus2: No parameter help available
- Cd_Aver: enums.ResultStatus2: No parameter help available
- Dc_Aver: enums.ResultStatus2: No parameter help available
- Ba_Aver: enums.ResultStatus2: No parameter help available
- Out_Of_Tol: enums.ResultStatus2: No parameter help available

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Ab_Aver: float: No parameter help available
- Cd_Aver: float: No parameter help available
- Dc_Aver: float: No parameter help available
- Ba_Aver: float: No parameter help available
- Out_Of_Tol: float: No parameter help available

class ReadStruct

Response structure. Fields:

- Reliability: str: No parameter help available
- Ab_Aver: float: No parameter help available
- Cd_Aver: float: No parameter help available
- Dc_Aver: float: No parameter help available
- Ba_Aver: float: No parameter help available

- Out_Of_Tol: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:DSSS:AVERage
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.dsss.average.
↪calculate()
```

No command help available

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:DSSS:AVERage
value: FetchStruct = driver.wlanMeas.multiEval.tsMask.dsss.average.fetch()
```

No command help available

return
structure: for return value, see the help for FetchStruct structure arguments.

read() → ReadStruct

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:DSSS:AVERage
value: ReadStruct = driver.wlanMeas.multiEval.tsMask.dsss.average.read()
```

No command help available

return
structure: for return value, see the help for ReadStruct structure arguments.

6.4.1.10.4.2 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:DSSS:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:DSSS:CURRent
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:DSSS:CURRent
```

class CurrentCls

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Ab_Curr: enums.ResultStatus2: No parameter help available
- Cd_Curr: enums.ResultStatus2: No parameter help available
- Dc_Curr: enums.ResultStatus2: No parameter help available
- Ba_Curr: enums.ResultStatus2: No parameter help available
- Out_Of_Tol: enums.ResultStatus2: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Ab_Curr: float: No parameter help available
- Cd_Curr: float: No parameter help available
- Dc_Curr: float: No parameter help available
- Ba_Curr: float: No parameter help available
- Out_Of_Tol: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:DSSS:CURRent
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.dsss.current.
↪calculate()
```

No command help available

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:DSSS:CURRent
value: ResultData = driver.wlanMeas.multiEval.tsMask.dsss.current.fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:DSSS:CURRent
value: ResultData = driver.wlanMeas.multiEval.tsMask.dsss.current.read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.10.4.3 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:DSSS:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:DSSS:MAXimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:DSSS:MAXimum
```

class MaximumCls

Maximum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Ab_Max: enums.ResultStatus2: No parameter help available
- Cd_Max: enums.ResultStatus2: No parameter help available
- Dcmax: enums.ResultStatus2: No parameter help available
- Ba_Max: enums.ResultStatus2: No parameter help available
- Out_Of_Tol: enums.ResultStatus2: No parameter help available

class FetchStruct

Response structure. Fields:

- Reliability: str: No parameter help available
- Ab_Max: float: No parameter help available
- Cd_Max: float: No parameter help available
- Dcmax: float: No parameter help available
- Ba_Max: float: No parameter help available
- Out_Of_Tol: float: No parameter help available

class ReadStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Ab_Max: float: No parameter help available
- Cd_Max: float: No parameter help available
- Dcmax: float: No parameter help available
- Ba_Max: float: No parameter help available
- Out_Of_Tol: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:DSSS:MAXimum
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.dsss.maximum.
↪calculate()
```

No command help available

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:DSSS:MAXimum
value: FetchStruct = driver.wlanMeas.multiEval.tsMask.dsss.maximum.fetch()
```

No command help available

return
structure: for return value, see the help for FetchStruct structure arguments.

read() → ReadStruct

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:DSSS:MAXimum
value: ReadStruct = driver.wlanMeas.multiEval.tsMask.dsss.maximum.read()
```

No command help available

return
structure: for return value, see the help for ReadStruct structure arguments.

6.4.1.10.5 Frequency

class FrequencyCls

Frequency commands group definition. 12 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.tsMask.frequency.clone()
```

Subgroups

6.4.1.10.5.1 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:FREQuency:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:FREQuency:AVERage
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:FREQuency:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tol: enums.ResultStatus2: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified transmit spectrum mask limits.
- Margin_Xvals: List[enums.ResultStatus2]: Comma-separated list of frequencies, one value per margin The number of margins equals the number of spectrum mask areas and depends on the selected standard, see Table 'Spectrum mask areas'.

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified transmit spectrum mask limits.

- Margin_Xvals: List[float]: Comma-separated list of frequencies, one value per margin The number of margins equals the number of spectrum mask areas and depends on the selected standard, see Table 'Spectrum mask areas'.

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>
↪:MEValuation:TSMask:FREQuency:AVERage
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.frequency.average.
↪calculate()
```

Return the X-positions of the limit line margins of the transmit spectrum mask. Positions for the current, average, minimum and maximum traces are returned. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:FREQuency:AVERage
value: ResultData = driver.wlanMeas.multiEval.tsMask.frequency.average.fetch()
```

Return the X-positions of the limit line margins of the transmit spectrum mask. Positions for the current, average, minimum and maximum traces are returned. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:FREQuency:AVERage
value: ResultData = driver.wlanMeas.multiEval.tsMask.frequency.average.read()
```

Return the X-positions of the limit line margins of the transmit spectrum mask. Positions for the current, average, minimum and maximum traces are returned. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.10.5.2 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:FREQuency:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:FREQuency:CURRent
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:FREQuency:CURRent
```

class CurrentCls

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tol: enums.ResultStatus2: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified transmit spectrum mask limits.
- Margin_Xvals: List[enums.ResultStatus2]: Comma-separated list of frequencies, one value per margin The number of margins equals the number of spectrum mask areas and depends on the selected standard, see Table 'Spectrum mask areas'.

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified transmit spectrum mask limits.
- Margin_Xvals: List[float]: Comma-separated list of frequencies, one value per margin The number of margins equals the number of spectrum mask areas and depends on the selected standard, see Table 'Spectrum mask areas'.

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>
↪:MEValuation:TSMask:FREQuency:CURRent
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.frequency.current.
↪calculate()
```

Return the X-positions of the limit line margins of the transmit spectrum mask. Positions for the current, average, minimum and maximum traces are returned. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:FREQuency:CURRent
value: ResultData = driver.wlanMeas.multiEval.tsMask.frequency.current.fetch()
```

Return the X-positions of the limit line margins of the transmit spectrum mask. Positions for the current, average, minimum and maximum traces are returned. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:FREQuency:CURRent
value: ResultData = driver.wlanMeas.multiEval.tsMask.frequency.current.read()
```

Return the X-positions of the limit line margins of the transmit spectrum mask. Positions for the current, average, minimum and maximum traces are returned. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.10.5.3 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:FREQuency:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:FREQuency:MAXimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:FREQuency:MAXimum
```

class MaximumCls

Maximum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tol: enums.ResultStatus2: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified transmit spectrum mask limits.
- Margin_Xvals: List[enums.ResultStatus2]: Comma-separated list of frequencies, one value per margin The number of margins equals the number of spectrum mask areas and depends on the selected standard, see Table 'Spectrum mask areas'.

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified transmit spectrum mask limits.
- Margin_Xvals: List[float]: Comma-separated list of frequencies, one value per margin The number of margins equals the number of spectrum mask areas and depends on the selected standard, see Table 'Spectrum mask areas'.

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>
↪:MEValuation:TSMask:FREQuency:MAXimum
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.frequency.maximum.
↪calculate()
```

Return the X-positions of the limit line margins of the transmit spectrum mask. Positions for the current, average, minimum and maximum traces are returned. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:FREQuency:MAXimum
value: ResultData = driver.wlanMeas.multiEval.tsMask.frequency.maximum.fetch()
```

Return the X-positions of the limit line margins of the transmit spectrum mask. Positions for the current, average, minimum and maximum traces are returned. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:FREQuency:MAXimum
value: ResultData = driver.wlanMeas.multiEval.tsMask.frequency.maximum.read()
```

Return the X-positions of the limit line margins of the transmit spectrum mask. Positions for the current, average, minimum and maximum traces are returned. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.10.5.4 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:FREQuency:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:FREQuency:MINimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:FREQuency:MINimum
```

class MinimumCls

Minimum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tol: enums.ResultStatus2: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified transmit spectrum mask limits.
- Margin_Xvals: List[enums.ResultStatus2]: Comma-separated list of frequencies, one value per margin The number of margins equals the number of spectrum mask areas and depends on the selected standard, see Table 'Spectrum mask areas'.

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified transmit spectrum mask limits.

- Margin_Xvals: List[float]: Comma-separated list of frequencies, one value per margin The number of margins equals the number of spectrum mask areas and depends on the selected standard, see Table 'Spectrum mask areas'.

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>
↪:MEValuation:TSMask:FREQuency:MINimum
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.frequency.minimum.
↪calculate()
```

Return the X-positions of the limit line margins of the transmit spectrum mask. Positions for the current, average, minimum and maximum traces are returned. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:FREQuency:MINimum
value: ResultData = driver.wlanMeas.multiEval.tsMask.frequency.minimum.fetch()
```

Return the X-positions of the limit line margins of the transmit spectrum mask. Positions for the current, average, minimum and maximum traces are returned. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:FREQuency:MINimum
value: ResultData = driver.wlanMeas.multiEval.tsMask.frequency.minimum.read()
```

Return the X-positions of the limit line margins of the transmit spectrum mask. Positions for the current, average, minimum and maximum traces are returned. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.10.6 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MAXimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MAXimum
```

class MaximumCls

Maximum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tol: enums.ResultStatus2: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified transmit spectrum mask limits.
- Margin: List[enums.ResultStatus2]: Comma-separated list of margin values, one value per spectrum mask area The number of margin values depends on the selected standard, see Table 'Spectrum mask areas'.

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified transmit spectrum mask limits.
- Margin: List[float]: Comma-separated list of margin values, one value per spectrum mask area The number of margin values depends on the selected standard, see Table 'Spectrum mask areas'.

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MAXimum
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.maximum.calculate()
```

Return the limit line margin values of the transmit spectrum mask for SISO measurements and bandwidths with one segment. Margins for the current, average, minimum and maximum traces are returned. A positive result indicates that the trace is located above the limit line. The limit is exceeded. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MAXimum
value: ResultData = driver.wlanMeas.multiEval.tsMask.maximum.fetch()
```

Return the limit line margin values of the transmit spectrum mask for SISO measurements and bandwidths with one segment. Margins for the current, average, minimum and maximum traces are returned. A positive result indicates that the trace is located above the limit line. The limit is exceeded. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:MAXimum
value: ResultData = driver.wlanMeas.multiEval.tsMask.maximum.read()
```

Return the limit line margin values of the transmit spectrum mask for SISO measurements and bandwidths with one segment. Margins for the current, average, minimum and maximum traces are returned. A positive

result indicates that the trace is located above the limit line. The limit is exceeded. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.10.7 Mimo<Mimo>

RepCap Settings

```
# Range: Nr1 .. Nr8
rc = driver.wlanMeas.multiEval.tsMask.mimo.repcap_mimo_get()
driver.wlanMeas.multiEval.tsMask.mimo.repcap_mimo_set(repcap.Mimo.Nr1)
```

class MimoCls

Mimo commands group definition. 48 total commands, 6 Subgroups, 0 group commands Repeated Capability: Mimo, default value after init: Mimo.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.tsMask.mimo.clone()
```

Subgroups

6.4.1.10.7.1 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:AVERage
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tol: enums.ResultStatus2: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified transmit spectrum mask limits.
- Margin_Tx: List[enums.ResultStatus2]: Comma-separated list of margin values, one value per spectrum mask area The number of margin values depends on the selected standard, see Table 'Spectrum mask areas'.

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified transmit spectrum mask limits.
- Margin_Tx: List[float]: Comma-separated list of margin values, one value per spectrum mask area The number of margin values depends on the selected standard, see Table 'Spectrum mask areas'.

calculate(*mimo=Mimo.Default*) → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:AVERage
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.mimo.average.
↪calculate(mimo = repcap.Mimo.Default)
```

Return the limit line margin values of the transmit spectrum mask for MIMO measurements, antenna <n>, bandwidths with one segment. Margins for the current, average, minimum and maximum traces are returned. A positive result indicates that the trace is located above the limit line. The limit is exceeded. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch(*mimo=Mimo.Default*) → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:AVERage
value: ResultData = driver.wlanMeas.multiEval.tsMask.mimo.average.fetch(mimo =␣
↪repcap.Mimo.Default)
```

Return the limit line margin values of the transmit spectrum mask for MIMO measurements, antenna <n>, bandwidths with one segment. Margins for the current, average, minimum and maximum traces are returned. A positive result indicates that the trace is located above the limit line. The limit is exceeded. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

read(*mimo=Mimo.Default*) → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:AVERage
value: ResultData = driver.wlanMeas.multiEval.tsMask.mimo.average.read(mimo =␣
↪repcap.Mimo.Default)
```

Return the limit line margin values of the transmit spectrum mask for MIMO measurements, antenna <n>, bandwidths with one segment. Margins for the current, average, minimum and maximum traces are returned. A positive result indicates that the trace is located above the limit line. The limit is exceeded. The

values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.10.7.2 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:CURRent
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:CURRent
```

class CurrentCls

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tol: enums.ResultStatus2: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified transmit spectrum mask limits.
- Margin_Tx: List[enums.ResultStatus2]: Comma-separated list of margin values, one value per spectrum mask area The number of margin values depends on the selected standard, see Table 'Spectrum mask areas'.

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified transmit spectrum mask limits.
- Margin_Tx: List[float]: Comma-separated list of margin values, one value per spectrum mask area The number of margin values depends on the selected standard, see Table 'Spectrum mask areas'.

calculate(*mimo=Mimo.Default*) → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:CURRent
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.mimo.current.
↪calculate(mimo = repcap.Mimo.Default)
```

Return the limit line margin values of the transmit spectrum mask for MIMO measurements, antenna <n>, bandwidths with one segment. Margins for the current, average, minimum and maximum traces are returned. A positive result indicates that the trace is located above the limit line. The limit is exceeded. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch(*mimo=Mimo.Default*) → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:CURRent
value: ResultData = driver.wlanMeas.multiEval.tsMask.mimo.current.fetch(mimo =
↪repcap.Mimo.Default)
```

Return the limit line margin values of the transmit spectrum mask for MIMO measurements, antenna <n>, bandwidths with one segment. Margins for the current, average, minimum and maximum traces are returned. A positive result indicates that the trace is located above the limit line. The limit is exceeded. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

read(*mimo=Mimo.Default*) → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:CURRent
value: ResultData = driver.wlanMeas.multiEval.tsMask.mimo.current.read(mimo =
↪repcap.Mimo.Default)
```

Return the limit line margin values of the transmit spectrum mask for MIMO measurements, antenna <n>, bandwidths with one segment. Margins for the current, average, minimum and maximum traces are returned. A positive result indicates that the trace is located above the limit line. The limit is exceeded. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.10.7.3 Frequency

class FrequencyCls

Frequency commands group definition. 12 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.tsMask.mimo.frequency.clone()
```

Subgroups

6.4.1.10.7.4 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:FREQuency:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:FREQuency:AVERage
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:FREQuency:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tol: enums.ResultStatus2: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified transmit spectrum mask limits.
- Margin_Xvals_Tx: List[enums.ResultStatus2]: Comma-separated list of frequencies, one value per margin The number of margins equals the number of spectrum mask areas and depends on the selected standard, see Table 'Spectrum mask areas'.

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified transmit spectrum mask limits.
- Margin_Xvals_Tx: List[float]: Comma-separated list of frequencies, one value per margin The number of margins equals the number of spectrum mask areas and depends on the selected standard, see Table 'Spectrum mask areas'.

calculate(*mimo=Mimo.Default*) → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>
↪:FREQuency:AVERage
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.mimo.frequency.
↪average.calculate(mimo = repcap.Mimo.Default)
```

Return the X-positions of the limit line margins of the transmit spectrum mask, for MIMO measurements, antenna <n>, bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch(*mimo=Mimo.Default*) → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>
↪:FREQuency:AVERage
value: ResultData = driver.wlanMeas.multiEval.tsMask.mimo.frequency.average.
↪fetch(mimo = repcap.Mimo.Default)
```

Return the X-positions of the limit line margins of the transmit spectrum mask, for MIMO measurements, antenna <n>, bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

read(*mimo=Mimo.Default*) → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>
↪:FREQuency:AVERage
value: ResultData = driver.wlanMeas.multiEval.tsMask.mimo.frequency.average.
↪read(mimo = repcap.Mimo.Default)
```

Return the X-positions of the limit line margins of the transmit spectrum mask, for MIMO measurements, antenna <n>, bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.10.7.5 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:FREQuency:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:FREQuency:CURRent
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:FREQuency:CURRent
```

class CurrentCls

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tol: enums.ResultStatus2: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified transmit spectrum mask limits.
- Margin_Xvals_Tx: List[enums.ResultStatus2]: Comma-separated list of frequencies, one value per margin The number of margins equals the number of spectrum mask areas and depends on the selected standard, see Table 'Spectrum mask areas'.

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified transmit spectrum mask limits.
- Margin_Xvals_Tx: List[float]: Comma-separated list of frequencies, one value per margin The number of margins equals the number of spectrum mask areas and depends on the selected standard, see Table 'Spectrum mask areas'.

calculate(*mimo=Mimo.Default*) → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>
↪:FREQuency:CURRent
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.mimo.frequency.
↪current.calculate(mimo = repcap.Mimo.Default)
```

Return the X-positions of the limit line margins of the transmit spectrum mask, for MIMO measurements, antenna <n>, bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch(*mimo=Mimo.Default*) → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>
↪:FREQuency:CURRent
value: ResultData = driver.wlanMeas.multiEval.tsMask.mimo.frequency.current.
↪fetch(mimo = repcap.Mimo.Default)
```

Return the X-positions of the limit line margins of the transmit spectrum mask, for MIMO measurements, antenna <n>, bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

read(*mimo=Mimo.Default*) → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>
↪:FREQuency:CURRent
value: ResultData = driver.wlanMeas.multiEval.tsMask.mimo.frequency.current.
↪read(mimo = repcap.Mimo.Default)
```

Return the X-positions of the limit line margins of the transmit spectrum mask, for MIMO measurements, antenna <n>, bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.10.7.6 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:FREQuency:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:FREQuency:MAXimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:FREQuency:MAXimum
```

class MaximumCls

Maximum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tol: enums.ResultStatus2: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified transmit spectrum mask limits.
- Margin_Xvals_Tx: List[enums.ResultStatus2]: Comma-separated list of frequencies, one value per margin The number of margins equals the number of spectrum mask areas and depends on the selected standard, see Table 'Spectrum mask areas'.

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified transmit spectrum mask limits.
- Margin_Xvals_Tx: List[float]: Comma-separated list of frequencies, one value per margin The number of margins equals the number of spectrum mask areas and depends on the selected standard, see Table 'Spectrum mask areas'.

calculate(*mimo=Mimo.Default*) → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>
↪:FREQuency:MAXimum
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.mimo.frequency.
↪maximum.calculate(mimo = repcap.Mimo.Default)
```

Return the X-positions of the limit line margins of the transmit spectrum mask, for MIMO measurements, antenna <n>, bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch(*mimo=Mimo.Default*) → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>
↪:FREQuency:MAXimum
value: ResultData = driver.wlanMeas.multiEval.tsMask.mimo.frequency.maximum.
↪fetch(mimo = repcap.Mimo.Default)
```

Return the X-positions of the limit line margins of the transmit spectrum mask, for MIMO measurements, antenna <n>, bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

read(*mimo=Mimo.Default*) → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>
↪:FREQuency:MAXimum
value: ResultData = driver.wlanMeas.multiEval.tsMask.mimo.frequency.maximum.
↪read(mimo = repcap.Mimo.Default)
```

Return the X-positions of the limit line margins of the transmit spectrum mask, for MIMO measurements, antenna <n>, bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.10.7.7 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:FREQuency:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:FREQuency:MINimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:FREQuency:MINimum
```

class MinimumCls

Minimum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tol: enums.ResultStatus2: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified transmit spectrum mask limits.
- Margin_Xvals_Tx: List[enums.ResultStatus2]: Comma-separated list of frequencies, one value per margin The number of margins equals the number of spectrum mask areas and depends on the selected standard, see Table 'Spectrum mask areas'.

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified transmit spectrum mask limits.
- Margin_Xvals_Tx: List[float]: Comma-separated list of frequencies, one value per margin The number of margins equals the number of spectrum mask areas and depends on the selected standard, see Table 'Spectrum mask areas'.

calculate(*mimo=Mimo.Default*) → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>
↪:FREQuency:MINimum
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.mimo.frequency.
↪minimum.calculate(mimo = repcap.Mimo.Default)
```

Return the X-positions of the limit line margins of the transmit spectrum mask, for MIMO measurements, antenna <n>, bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch(*mimo=Mimo.Default*) → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>
↪:FREQuency:MINimum
value: ResultData = driver.wlanMeas.multiEval.tsMask.mimo.frequency.minimum.
↪fetch(mimo = repcap.Mimo.Default)
```

Return the X-positions of the limit line margins of the transmit spectrum mask, for MIMO measurements, antenna <n>, bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

read(*mimo=Mimo.Default*) → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>
↪:FREQuency:MINimum
value: ResultData = driver.wlanMeas.multiEval.tsMask.mimo.frequency.minimum.
↪read(mimo = repcap.Mimo.Default)
```

Return the X-positions of the limit line margins of the transmit spectrum mask, for MIMO measurements, antenna <n>, bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.10.7.8 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:MAXimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:MAXimum
```

class MaximumCls

Maximum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tol: enums.ResultStatus2: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified transmit spectrum mask limits.

- Margin_Tx: List[enums.ResultStatus2]: Comma-separated list of margin values, one value per spectrum mask area The number of margin values depends on the selected standard, see Table 'Spectrum mask areas'.

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified transmit spectrum mask limits.
- Margin_Tx: List[float]: Comma-separated list of margin values, one value per spectrum mask area The number of margin values depends on the selected standard, see Table 'Spectrum mask areas'.

calculate(*mimo=Mimo.Default*) → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:MAXimum
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.mimo.maximum.
↪calculate(mimo = repcap.Mimo.Default)
```

Return the limit line margin values of the transmit spectrum mask for MIMO measurements, antenna <n>, bandwidths with one segment. Margins for the current, average, minimum and maximum traces are returned. A positive result indicates that the trace is located above the limit line. The limit is exceeded. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch(*mimo=Mimo.Default*) → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:MAXimum
value: ResultData = driver.wlanMeas.multiEval.tsMask.mimo.maximum.fetch(mimo =␣
↪repcap.Mimo.Default)
```

Return the limit line margin values of the transmit spectrum mask for MIMO measurements, antenna <n>, bandwidths with one segment. Margins for the current, average, minimum and maximum traces are returned. A positive result indicates that the trace is located above the limit line. The limit is exceeded. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

read(*mimo=Mimo.Default*) → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:MAXimum
value: ResultData = driver.wlanMeas.multiEval.tsMask.mimo.maximum.read(mimo =␣
↪repcap.Mimo.Default)
```

Return the limit line margin values of the transmit spectrum mask for MIMO measurements, antenna <n>, bandwidths with one segment. Margins for the current, average, minimum and maximum traces are returned. A positive result indicates that the trace is located above the limit line. The limit is exceeded. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.10.7.9 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:MINimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:MINimum
```

class MinimumCls

Minimum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tol: enums.ResultStatus2: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified transmit spectrum mask limits.
- Margin_Tx: List[enums.ResultStatus2]: Comma-separated list of margin values, one value per spectrum mask area The number of margin values depends on the selected standard, see Table 'Spectrum mask areas'.

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified transmit spectrum mask limits.
- Margin_Tx: List[float]: Comma-separated list of margin values, one value per spectrum mask area The number of margin values depends on the selected standard, see Table 'Spectrum mask areas'.

calculate(*mimo=Mimo.Default*) → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:MINimum
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.mimo.minimum.
↪calculate(mimo = repcap.Mimo.Default)
```

Return the limit line margin values of the transmit spectrum mask for MIMO measurements, antenna <n>, bandwidths with one segment. Margins for the current, average, minimum and maximum traces are returned. A positive result indicates that the trace is located above the limit line. The limit is exceeded. The

values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch(*mimo=Mimo.Default*) → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:MINimum
value: ResultData = driver.wlanMeas.multiEval.tsMask.mimo.minimum.fetch(mimo =␣
↪repcap.Mimo.Default)
```

Return the limit line margin values of the transmit spectrum mask for MIMO measurements, antenna <n>, bandwidths with one segment. Margins for the current, average, minimum and maximum traces are returned. A positive result indicates that the trace is located above the limit line. The limit is exceeded. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

read(*mimo=Mimo.Default*) → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:MINimum
value: ResultData = driver.wlanMeas.multiEval.tsMask.mimo.minimum.read(mimo =␣
↪repcap.Mimo.Default)
```

Return the limit line margin values of the transmit spectrum mask for MIMO measurements, antenna <n>, bandwidths with one segment. Margins for the current, average, minimum and maximum traces are returned. A positive result indicates that the trace is located above the limit line. The limit is exceeded. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.10.7.10 Segments

class SegmentsCls

Segments commands group definition. 24 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.tsMask.mimo.segments.clone()
```

Subgroups

6.4.1.10.7.11 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:SEGMents:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:SEGMents:AVERage
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:SEGMents:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tol_Seg_1: enums.ResultStatus2: No parameter help available
- Out_Of_Tol_Seg_2: enums.ResultStatus2: No parameter help available
- Margin_Seg_1_Tx: List[enums.ResultStatus2]: No parameter help available
- Margin_Seg_2_Tx: List[enums.ResultStatus2]: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tol_Seg_1: float: No parameter help available
- Out_Of_Tol_Seg_2: float: No parameter help available
- Margin_Seg_1_Tx: List[float]: No parameter help available
- Margin_Seg_2_Tx: List[float]: No parameter help available

calculate(*mimo=Mimo.Default*) → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>
↪:SEGMents:AVERage
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.mimo.segments.average.
↪calculate(mimo = repcap.Mimo.Default)
```

No command help available

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch(*mimo=Mimo.Default*) → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>
↪:SEGMents:AVERage
value: ResultData = driver.wlanMeas.multiEval.tsMask.mimo.segments.average.
↪fetch(mimo = repcap.Mimo.Default)
```

No command help available

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

read(*mimo=Mimo.Default*) → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>
↪:SEGMents:AVERage
value: ResultData = driver.wlanMeas.multiEval.tsMask.mimo.segments.average.
↪read(mimo = repcap.Mimo.Default)
```

No command help available

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.10.7.12 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:SEGMents:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:SEGMents:CURRent
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:SEGMents:CURRent
```

class CurrentCls

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tol_Seg_1: enums.ResultStatus2: No parameter help available

- Out_Of_Tol_Seg_2: enums.ResultStatus2: No parameter help available
- Margin_Seg_1_Tx: List[enums.ResultStatus2]: No parameter help available
- Margin_Seg_2_Tx: List[enums.ResultStatus2]: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tol_Seg_1: float: No parameter help available
- Out_Of_Tol_Seg_2: float: No parameter help available
- Margin_Seg_1_Tx: List[float]: No parameter help available
- Margin_Seg_2_Tx: List[float]: No parameter help available

calculate(*mimo=Mimo.Default*) → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>
↪:SEGMents:CURRent
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.mimo.segments.current.
↪calculate(mimo = repcap.Mimo.Default)
```

No command help available

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch(*mimo=Mimo.Default*) → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>
↪:SEGMents:CURRent
value: ResultData = driver.wlanMeas.multiEval.tsMask.mimo.segments.current.
↪fetch(mimo = repcap.Mimo.Default)
```

No command help available

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

read(*mimo=Mimo.Default*) → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>
↪:SEGMents:CURRent
value: ResultData = driver.wlanMeas.multiEval.tsMask.mimo.segments.current.
↪read(mimo = repcap.Mimo.Default)
```

No command help available

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.10.7.13 Frequency

class FrequencyCls

Frequency commands group definition. 12 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.tsMask.mimo.segments.frequency.clone()
```

Subgroups

6.4.1.10.7.14 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:SEGMents:FREQuency:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:SEGMents:FREQuency:AVERage
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>
↪:SEGMents:FREQuency:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tol_Seg_1: enums.ResultStatus2: No parameter help available
- Out_Of_Tol_Seg_2: enums.ResultStatus2: No parameter help available
- Margin_Xvals_Seg_1_Tx: List[enums.ResultStatus2]: No parameter help available
- Margin_Xvals_Seg_2_Tx: List[enums.ResultStatus2]: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tol_Seg_1: float: No parameter help available
- Out_Of_Tol_Seg_2: float: No parameter help available
- Margin_Xvals_Seg_1_Tx: List[float]: No parameter help available
- Margin_Xvals_Seg_2_Tx: List[float]: No parameter help available

calculate(*mimo=Mimo.Default*) → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>
↪:SEGMents:FREQuency:AVERage
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.mimo.segments.
↪frequency.average.calculate(mimo = repcap.Mimo.Default)
```

No command help available

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch(*mimo=Mimo.Default*) → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>
↪:SEGMents:FREQuency:AVERage
value: ResultData = driver.wlanMeas.multiEval.tsMask.mimo.segments.frequency.
↪average.fetch(mimo = repcap.Mimo.Default)
```

No command help available

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

read(*mimo=Mimo.Default*) → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>
↪:SEGMents:FREQuency:AVERage
value: ResultData = driver.wlanMeas.multiEval.tsMask.mimo.segments.frequency.
↪average.read(mimo = repcap.Mimo.Default)
```

No command help available

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.10.7.15 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:SEGMents:FREQuency:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:SEGMents:FREQuency:CURRent
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>
↪:SEGMents:FREQuency:CURRent
```

class CurrentCls

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tol_Seg_1: enums.ResultStatus2: No parameter help available
- Out_Of_Tol_Seg_2: enums.ResultStatus2: No parameter help available
- Margin_Xvals_Seg_1_Tx: List[enums.ResultStatus2]: No parameter help available
- Margin_Xvals_Seg_2_Tx: List[enums.ResultStatus2]: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tol_Seg_1: float: No parameter help available
- Out_Of_Tol_Seg_2: float: No parameter help available
- Margin_Xvals_Seg_1_Tx: List[float]: No parameter help available
- Margin_Xvals_Seg_2_Tx: List[float]: No parameter help available

calculate(*mimo=Mimo.Default*) → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>
↪:SEGMents:FREQuency:CURRent
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.mimo.segments.
↪frequency.current.calculate(mimo = repcap.Mimo.Default)
```

No command help available

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch(*mimo=Mimo.Default*) → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>
↪:SEGMents:FREQuency:CURRent
value: ResultData = driver.wlanMeas.multiEval.tsMask.mimo.segments.frequency.
↪current.fetch(mimo = repcap.Mimo.Default)
```

No command help available

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

read(*mimo=Mimo.Default*) → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>
↪:SEGMents:FREQuency:CURRent
value: ResultData = driver.wlanMeas.multiEval.tsMask.mimo.segments.frequency.
↪current.read(mimo = repcap.Mimo.Default)
```

No command help available

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.10.7.16 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:SEGMents:FREQuency:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:SEGMents:FREQuency:MAXimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>
↪:SEGMents:FREQuency:MAXimum
```

class MaximumCls

Maximum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tol_Seg_1: enums.ResultStatus2: No parameter help available
- Out_Of_Tol_Seg_2: enums.ResultStatus2: No parameter help available
- Margin_Xvals_Seg_1_Tx: List[enums.ResultStatus2]: No parameter help available
- Margin_Xvals_Seg_2_Tx: List[enums.ResultStatus2]: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tol_Seg_1: float: No parameter help available
- Out_Of_Tol_Seg_2: float: No parameter help available
- Margin_Xvals_Seg_1_Tx: List[float]: No parameter help available

- Margin_Xvals_Seg_2_Tx: List[float]: No parameter help available

calculate(*mimo=Mimo.Default*) → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>
↪:SEGMents:FREQuency:MAXimum
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.mimo.segments.
↪frequency.maximum.calculate(mimo = repcap.Mimo.Default)
```

No command help available

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch(*mimo=Mimo.Default*) → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>
↪:SEGMents:FREQuency:MAXimum
value: ResultData = driver.wlanMeas.multiEval.tsMask.mimo.segments.frequency.
↪maximum.fetch(mimo = repcap.Mimo.Default)
```

No command help available

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

read(*mimo=Mimo.Default*) → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>
↪:SEGMents:FREQuency:MAXimum
value: ResultData = driver.wlanMeas.multiEval.tsMask.mimo.segments.frequency.
↪maximum.read(mimo = repcap.Mimo.Default)
```

No command help available

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.10.7.17 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:SEGMents:FREQuency:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:SEGMents:FREQuency:MINimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>
↪:SEGMents:FREQuency:MINimum
```

class MinimumCls

Minimum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tol_Seg_1: enums.ResultStatus2: No parameter help available
- Out_Of_Tol_Seg_2: enums.ResultStatus2: No parameter help available
- Margin_Xvals_Seg_1_Tx: List[enums.ResultStatus2]: No parameter help available
- Margin_Xvals_Seg_2_Tx: List[enums.ResultStatus2]: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tol_Seg_1: float: No parameter help available
- Out_Of_Tol_Seg_2: float: No parameter help available
- Margin_Xvals_Seg_1_Tx: List[float]: No parameter help available
- Margin_Xvals_Seg_2_Tx: List[float]: No parameter help available

calculate(*mimo=Mimo.Default*) → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>
↪:SEGMents:FREQuency:MINimum
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.mimo.segments.
↪frequency.minimum.calculate(mimo = repcap.Mimo.Default)
```

No command help available

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch(*mimo=Mimo.Default*) → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>
↪:SEGMents:FREQuency:MINimum
value: ResultData = driver.wlanMeas.multiEval.tsMask.mimo.segments.frequency.
↪minimum.fetch(mimo = repcap.Mimo.Default)
```

No command help available

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

read(*mimo=Mimo.Default*) → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>
↪:SEGMents:FREQuency:MINimum
value: ResultData = driver.wlanMeas.multiEval.tsMask.mimo.segments.frequency.
↪minimum.read(mimo = repcap.Mimo.Default)
```

No command help available

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.10.7.18 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:SEGMents:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:SEGMents:MAXimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:SEGMents:MAXimum
```

class MaximumCls

Maximum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tol_Seg_1: enums.ResultStatus2: No parameter help available
- Out_Of_Tol_Seg_2: enums.ResultStatus2: No parameter help available
- Margin_Seg_1_Tx: List[enums.ResultStatus2]: No parameter help available
- Margin_Seg_2_Tx: List[enums.ResultStatus2]: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tol_Seg_1: float: No parameter help available
- Out_Of_Tol_Seg_2: float: No parameter help available
- Margin_Seg_1_Tx: List[float]: No parameter help available

- Margin_Seg_2_Tx: List[float]: No parameter help available

calculate(*mimo=Mimo.Default*) → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>
↪:SEGMents:MAXimum
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.mimo.segments.maximum.
↪calculate(mimo = repcap.Mimo.Default)
```

No command help available

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch(*mimo=Mimo.Default*) → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>
↪:SEGMents:MAXimum
value: ResultData = driver.wlanMeas.multiEval.tsMask.mimo.segments.maximum.
↪fetch(mimo = repcap.Mimo.Default)
```

No command help available

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

read(*mimo=Mimo.Default*) → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>
↪:SEGMents:MAXimum
value: ResultData = driver.wlanMeas.multiEval.tsMask.mimo.segments.maximum.
↪read(mimo = repcap.Mimo.Default)
```

No command help available

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.10.7.19 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:SEGMents:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:SEGMents:MINimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>:SEGMents:MINimum
```

class MinimumCls

Minimum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tol_Seg_1: enums.ResultStatus2: No parameter help available
- Out_Of_Tol_Seg_2: enums.ResultStatus2: No parameter help available
- Margin_Seg_1_Tx: List[enums.ResultStatus2]: No parameter help available
- Margin_Seg_2_Tx: List[enums.ResultStatus2]: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tol_Seg_1: float: No parameter help available
- Out_Of_Tol_Seg_2: float: No parameter help available
- Margin_Seg_1_Tx: List[float]: No parameter help available
- Margin_Seg_2_Tx: List[float]: No parameter help available

calculate(*mimo=Mimo.Default*) → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>
↪:SEGMents:MINimum
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.mimo.segments.minimum.
↪calculate(mimo = repcap.Mimo.Default)
```

No command help available

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch(*mimo=Mimo.Default*) → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>
↪:SEGMents:MINimum
value: ResultData = driver.wlanMeas.multiEval.tsMask.mimo.segments.minimum.
↪fetch(mimo = repcap.Mimo.Default)
```

No command help available

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

read(*mimo=Mimo.Default*) → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n>
↪:SEGMents:MINimum
value: ResultData = driver.wlanMeas.multiEval.tsMask.mimo.segments.minimum.
↪read(mimo = repcap.Mimo.Default)
```

No command help available

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.10.8 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MINimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MINimum
```

class MinimumCls

Minimum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tol: enums.ResultStatus2: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified transmit spectrum mask limits.
- Margin: List[enums.ResultStatus2]: Comma-separated list of margin values, one value per spectrum mask area The number of margin values depends on the selected standard, see Table 'Spectrum mask areas'.

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tol: float: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified transmit spectrum mask limits.
- Margin: List[float]: Comma-separated list of margin values, one value per spectrum mask area The number of margin values depends on the selected standard, see Table 'Spectrum mask areas'.

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MINimum
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.minimum.calculate()
```

Return the limit line margin values of the transmit spectrum mask for SISO measurements and bandwidths with one segment. Margins for the current, average, minimum and maximum traces are returned. A positive result indicates that the trace is located above the limit line. The limit is exceeded. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MINimum
value: ResultData = driver.wlanMeas.multiEval.tsMask.minimum.fetch()
```

Return the limit line margin values of the transmit spectrum mask for SISO measurements and bandwidths with one segment. Margins for the current, average, minimum and maximum traces are returned. A positive result indicates that the trace is located above the limit line. The limit is exceeded. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:MINimum
value: ResultData = driver.wlanMeas.multiEval.tsMask.minimum.read()
```

Return the limit line margin values of the transmit spectrum mask for SISO measurements and bandwidths with one segment. Margins for the current, average, minimum and maximum traces are returned. A positive result indicates that the trace is located above the limit line. The limit is exceeded. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.10.9 Nsiso

class NsisoCls

Nsiso commands group definition. 9 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.tsMask.nsiso.clone()
```

Subgroups

6.4.1.10.9.1 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:NSISo:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:NSISo:AVERage
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:NSISo:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Ab_Aver: enums.ResultStatus2: No parameter help available
- Bc_Aver: enums.ResultStatus2: No parameter help available
- Cd_Aver: enums.ResultStatus2: No parameter help available
- De_Aver: enums.ResultStatus2: No parameter help available
- Ed_Aver: enums.ResultStatus2: No parameter help available
- Dc_Aver: enums.ResultStatus2: No parameter help available
- Cb_Aver: enums.ResultStatus2: No parameter help available
- Ba_Aver: enums.ResultStatus2: No parameter help available
- Out_Of_Tol: enums.ResultStatus2: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Ab_Aver: float: No parameter help available
- Bc_Aver: float: No parameter help available
- Cd_Aver: float: No parameter help available
- De_Aver: float: No parameter help available
- Ed_Aver: float: No parameter help available
- Dc_Aver: float: No parameter help available
- Cb_Aver: float: No parameter help available
- Ba_Aver: float: No parameter help available

- Out_Of_Tol: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:NSISo:AVERage
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.nsiso.average.
↪calculate()
```

No command help available

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:NSISo:AVERage
value: ResultData = driver.wlanMeas.multiEval.tsMask.nsiso.average.fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:NSISo:AVERage
value: ResultData = driver.wlanMeas.multiEval.tsMask.nsiso.average.read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.10.9.2 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:NSISo:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:NSISo:CURRent
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:NSISo:CURRent
```

class CurrentCls

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Ab_Curr: enums.ResultStatus2: No parameter help available
- Bc_Curr: enums.ResultStatus2: No parameter help available
- Cd_Curr: enums.ResultStatus2: No parameter help available
- De_Curr: enums.ResultStatus2: No parameter help available
- Ed_Curr: enums.ResultStatus2: No parameter help available

- Dc_Curr: enums.ResultStatus2: No parameter help available
- Cb_Curr: enums.ResultStatus2: No parameter help available
- Ba_Curr: enums.ResultStatus2: No parameter help available
- Out_Of_Tol: enums.ResultStatus2: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Ab_Curr: float: No parameter help available
- Bc_Curr: float: No parameter help available
- Cd_Curr: float: No parameter help available
- De_Curr: float: No parameter help available
- Ed_Curr: float: No parameter help available
- Dc_Curr: float: No parameter help available
- Cb_Curr: float: No parameter help available
- Ba_Curr: float: No parameter help available
- Out_Of_Tol: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:NSISo:CURRent
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.nsiso.current.
↪calculate()
```

No command help available

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:NSISo:CURRent
value: ResultData = driver.wlanMeas.multiEval.tsMask.nsiso.current.fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:NSISo:CURRent
value: ResultData = driver.wlanMeas.multiEval.tsMask.nsiso.current.read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.10.9.3 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:NSISo:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:NSISo:MAXimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:NSISo:MAXimum
```

class MaximumCls

Maximum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Ab_Max: enums.ResultStatus2: No parameter help available
- Bc_Max: enums.ResultStatus2: No parameter help available
- Cd_Max: enums.ResultStatus2: No parameter help available
- De_Max: enums.ResultStatus2: No parameter help available
- Ed_Max: enums.ResultStatus2: No parameter help available
- Dcmax: enums.ResultStatus2: No parameter help available
- Cb_Max: enums.ResultStatus2: No parameter help available
- Ba_Max: enums.ResultStatus2: No parameter help available
- Out_Of_Tol: enums.ResultStatus2: No parameter help available

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Ab_Max: float: No parameter help available
- Bc_Max: float: No parameter help available
- Cd_Max: float: No parameter help available
- De_Max: float: No parameter help available
- Ed_Max: float: No parameter help available
- Dcmax: float: No parameter help available
- Cb_Max: float: No parameter help available
- Ba_Max: float: No parameter help available
- Out_Of_Tol: float: No parameter help available

class ReadStruct

Response structure. Fields:

- Reliability: bool: No parameter help available
- Ab_Max: float: No parameter help available
- Bc_Max: float: No parameter help available

- Cd_Max: float: No parameter help available
- De_Max: float: No parameter help available
- Ed_Max: float: No parameter help available
- Dcmax: float: No parameter help available
- Cb_Max: float: No parameter help available
- Ba_Max: float: No parameter help available
- Out_Of_Tol: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:NSISo:MAXimum
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.nsiso.maximum.
↪calculate()
```

No command help available

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → FetchStruct

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:NSISo:MAXimum
value: FetchStruct = driver.wlanMeas.multiEval.tsMask.nsiso.maximum.fetch()
```

No command help available

return
structure: for return value, see the help for FetchStruct structure arguments.

read() → ReadStruct

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:NSISo:MAXimum
value: ReadStruct = driver.wlanMeas.multiEval.tsMask.nsiso.maximum.read()
```

No command help available

return
structure: for return value, see the help for ReadStruct structure arguments.

6.4.1.10.10 Obw

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:OBW
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:OBW
```

class ObwCls

Obw commands group definition. 8 total commands, 2 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'

- Obw_Values: List[float]: No parameter help available
- Obw_Lr: List[float]: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:OBW
value: ResultData = driver.wlanMeas.multiEval.tsMask.obw.fetch()
```

Return the OBW results for SISO measurements and bandwidths with one segment.

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:OBW
value: ResultData = driver.wlanMeas.multiEval.tsMask.obw.read()
```

Return the OBW results for SISO measurements and bandwidths with one segment.

return
structure: for return value, see the help for ResultData structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.tsMask.obw.clone()
```

Subgroups

6.4.1.10.10.1 Mimo<Mimo>

RepCap Settings

```
# Range: Nr1 .. Nr8
rc = driver.wlanMeas.multiEval.tsMask.obw.mimo.repcap_mimo_get()
driver.wlanMeas.multiEval.tsMask.obw.mimo.repcap_mimo_set(repcap.Mimo.Nr1)
```

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:OBW:MIMO<n>
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:OBW:MIMO<n>
```

class MimoCls

Mimo commands group definition. 4 total commands, 1 Subgroups, 2 group commands Repeated Capability: Mimo, default value after init: Mimo.Nr1

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'

- Obw_Values_Tx: List[float]: No parameter help available
- Obw_Leri_Tx: List[float]: No parameter help available

fetch(*mimo=Mimo.Default*) → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:OBW:MIMO<n>
value: ResultData = driver.wlanMeas.multiEval.tsMask.obw.mimo.fetch(mimo =␣
↪repcap.Mimo.Default)
```

Return the OBW results for MIMO measurements, antenna <n>, bandwidths with one segment.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

read(*mimo=Mimo.Default*) → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:OBW:MIMO<n>
value: ResultData = driver.wlanMeas.multiEval.tsMask.obw.mimo.read(mimo =␣
↪repcap.Mimo.Default)
```

Return the OBW results for MIMO measurements, antenna <n>, bandwidths with one segment.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.tsMask.obw.mimo.clone()
```

Subgroups

6.4.1.10.10.2 Segments

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:OBW:MIMO<n>:SEGMents
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:OBW:MIMO<n>:SEGMents
```

class SegmentsCls

Segments commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'

- Obw_Values_Seg_1_Tx: List[float]: No parameter help available
- Obw_Values_Seg_2_Tx: List[float]: No parameter help available
- Obw_Leri_Seg_1_Tx: List[float]: No parameter help available
- Obw_Leri_Seg_2_Tx: List[float]: No parameter help available

fetch(*mimo=Mimo.Default*) → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:OBW:MIMO<n>:SEGMents
value: ResultData = driver.wlanMeas.multiEval.tsMask.obw.mimo.segments.
↪fetch(mimo = repcap.Mimo.Default)
```

Return the OBW results for MIMO measurements, antenna number <n>, bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

read(*mimo=Mimo.Default*) → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:OBW:MIMO<n>:SEGMents
value: ResultData = driver.wlanMeas.multiEval.tsMask.obw.mimo.segments.
↪read(mimo = repcap.Mimo.Default)
```

Return the OBW results for MIMO measurements, antenna number <n>, bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>.

param mimo
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mimo')

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.10.10.3 Segments

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:OBW:SEGMents
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:OBW:SEGMents
```

class SegmentsCls

Segments commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Obw_Values_Seg_1: List[float]: No parameter help available
- Obw_Values_Seg_2: List[float]: No parameter help available

- Obw_Lr_Seg_1: List[float]: No parameter help available
- Obw_Lr_Seg_2: List[float]: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:OBW:SEGMents
value: ResultData = driver.wlanMeas.multiEval.tsMask.obw.segments.fetch()
```

Return the OBW results for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>.

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:OBW:SEGMents
value: ResultData = driver.wlanMeas.multiEval.tsMask.obw.segments.read()
```

Return the OBW results for SISO measurements and bandwidths > 160 MHz. The results are available for the left 160 MHz segment <1> and for the right 160 MHz segment <2>.

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.10.11 Ofdm

class OfdmCls

Ofdm commands group definition. 9 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.tsMask.ofdm.clone()
```

Subgroups

6.4.1.10.11.1 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:OFDM:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:OFDM:AVERage
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:OFDM:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available

- Margins_11_Ag: List[enums.ResultStatus2]: No parameter help available
- Margins_11_P: List[enums.ResultStatus2]: No parameter help available
- Margins_11_Parib: List[float]: No parameter help available
- Out_Of_Tol: enums.ResultStatus2: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Margins_11_Ag: List[float]: No parameter help available
- Margins_11_P: List[float]: No parameter help available
- Margins_11_Parib: List[float]: No parameter help available
- Out_Of_Tol: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:OFDM:AVERage
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.ofdm.average.
↪calculate()
```

No command help available

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:OFDM:AVERage
value: ResultData = driver.wlanMeas.multiEval.tsMask.ofdm.average.fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:OFDM:AVERage
value: ResultData = driver.wlanMeas.multiEval.tsMask.ofdm.average.read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.10.11.2 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:OFDM:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:OFDM:CURRent
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:OFDM:CURRent
```

class CurrentCls

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Margins_11_Ag: List[enums.ResultStatus2]: No parameter help available
- Margins_11_P: List[enums.ResultStatus2]: No parameter help available
- Margins_11_Parib: List[float]: No parameter help available
- Out_Of_Tol: enums.ResultStatus2: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Margins_11_Ag: List[float]: No parameter help available
- Margins_11_P: List[float]: No parameter help available
- Margins_11_Parib: List[float]: No parameter help available
- Out_Of_Tol: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:OFDM:CURRent
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.ofdm.current.
↪calculate()
```

No command help available

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:OFDM:CURRent
value: ResultData = driver.wlanMeas.multiEval.tsMask.ofdm.current.fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:OFDM:CURRent
value: ResultData = driver.wlanMeas.multiEval.tsMask.ofdm.current.read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.10.11.3 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:OFDM:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:OFDM:MAXimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:OFDM:MAXimum
```

class MaximumCls

Maximum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Margins_11_Ag: List[enums.ResultStatus2]: No parameter help available
- Margins_11_P: List[enums.ResultStatus2]: No parameter help available
- Margins_11_Parib: List[float]: No parameter help available
- Out_Of_Tol: enums.ResultStatus2: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Margins_11_Ag: List[float]: No parameter help available
- Margins_11_P: List[float]: No parameter help available
- Margins_11_Parib: List[float]: No parameter help available
- Out_Of_Tol: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:OFDM:MAXimum
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.ofdm.maximum.
↪calculate()
```

No command help available

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:OFDM:MAXimum
value: ResultData = driver.wlanMeas.multiEval.tsMask.ofdm.maximum.fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:OFDM:MAXimum
value: ResultData = driver.wlanMeas.multiEval.tsMask.ofdm.maximum.read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.10.12 Segments

class SegmentsCls

Segments commands group definition. 24 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.tsMask.segments.clone()
```

Subgroups

6.4.1.10.12.1 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:AVERage
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tol_Seg_1: enums.ResultStatus2: No parameter help available
- Out_Of_Tol_Seg_2: enums.ResultStatus2: No parameter help available
- Margin_Seg_1: List[enums.ResultStatus2]: No parameter help available

- Margin_Seg_2: List[enums.ResultStatus2]: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tol_Seg_1: float: No parameter help available
- Out_Of_Tol_Seg_2: float: No parameter help available
- Margin_Seg_1: List[float]: No parameter help available
- Margin_Seg_2: List[float]: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:AVERage
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.segments.average.
↪calculate()
```

No command help available

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:AVERage
value: ResultData = driver.wlanMeas.multiEval.tsMask.segments.average.fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:AVERage
value: ResultData = driver.wlanMeas.multiEval.tsMask.segments.average.read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.10.12.2 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:CURRent
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:CURRent
```

class CurrentCls

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tol_Seg_1: enums.ResultStatus2: No parameter help available
- Out_Of_Tol_Seg_2: enums.ResultStatus2: No parameter help available
- Margin_Seg_1: List[enums.ResultStatus2]: No parameter help available
- Margin_Seg_2: List[enums.ResultStatus2]: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tol_Seg_1: float: No parameter help available
- Out_Of_Tol_Seg_2: float: No parameter help available
- Margin_Seg_1: List[float]: No parameter help available
- Margin_Seg_2: List[float]: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:CURRent
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.segments.current.
↪calculate()
```

No command help available

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:CURRent
value: ResultData = driver.wlanMeas.multiEval.tsMask.segments.current.fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:CURRent
value: ResultData = driver.wlanMeas.multiEval.tsMask.segments.current.read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.10.12.3 Frequency

class FrequencyCls

Frequency commands group definition. 12 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.tsMask.segments.frequency.clone()
```

Subgroups

6.4.1.10.12.4 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:FREQuency:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:FREQuency:AVERage
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:FREQuency:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tol_Seg_1: enums.ResultStatus2: No parameter help available
- Out_Of_Tol_Seg_2: enums.ResultStatus2: No parameter help available
- Margin_Xvals_Seg_1: List[enums.ResultStatus2]: No parameter help available
- Margin_Xvals_Seg_2: List[enums.ResultStatus2]: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tol_Seg_1: float: No parameter help available
- Out_Of_Tol_Seg_2: float: No parameter help available
- Margin_Xvals_Seg_1: List[float]: No parameter help available
- Margin_Xvals_Seg_2: List[float]: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>
↪:MEValuation:TSMask:SEGMents:FREQuency:AVERage
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.segments.frequency.
↪average.calculate()
```

No command help available

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TSMask:SEGMents:FREQuency:AVERage
value: ResultData = driver.wlanMeas.multiEval.tsMask.segments.frequency.average.
↪fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TSMask:SEGMents:FREQuency:AVERage
value: ResultData = driver.wlanMeas.multiEval.tsMask.segments.frequency.average.
↪read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.10.12.5 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:FREQuency:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:FREQuency:CURRent
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:FREQuency:CURRent
```

class CurrentCls

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tol_Seg_1: enums.ResultStatus2: No parameter help available
- Out_Of_Tol_Seg_2: enums.ResultStatus2: No parameter help available
- Margin_Xvals_Seg_1: List[enums.ResultStatus2]: No parameter help available
- Margin_Xvals_Seg_2: List[enums.ResultStatus2]: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tol_Seg_1: float: No parameter help available

- Out_Of_Tol_Seg_2: float: No parameter help available
- Margin_Xvals_Seg_1: List[float]: No parameter help available
- Margin_Xvals_Seg_2: List[float]: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>
↪:MEValuation:TSMask:SEGMents:FREQuency:CURRent
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.segments.frequency.
↪current.calculate()
```

No command help available

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TSMask:SEGMents:FREQuency:CURRent
value: ResultData = driver.wlanMeas.multiEval.tsMask.segments.frequency.current.
↪fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TSMask:SEGMents:FREQuency:CURRent
value: ResultData = driver.wlanMeas.multiEval.tsMask.segments.frequency.current.
↪read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.10.12.6 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:FREQuency:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:FREQuency:MAXimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:FREQuency:MAXimum
```

class MaximumCls

Maximum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available

- Out_Of_Tol_Seg_1: enums.ResultStatus2: No parameter help available
- Out_Of_Tol_Seg_2: enums.ResultStatus2: No parameter help available
- Margin_Xvals_Seg_1: List[enums.ResultStatus2]: No parameter help available
- Margin_Xvals_Seg_2: List[enums.ResultStatus2]: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tol_Seg_1: float: No parameter help available
- Out_Of_Tol_Seg_2: float: No parameter help available
- Margin_Xvals_Seg_1: List[float]: No parameter help available
- Margin_Xvals_Seg_2: List[float]: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>
↪:MEValuation:TSMask:SEGMents:FREQuency:MAXimum
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.segments.frequency.
↪maximum.calculate()
```

No command help available

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TSMask:SEGMents:FREQuency:MAXimum
value: ResultData = driver.wlanMeas.multiEval.tsMask.segments.frequency.maximum.
↪fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TSMask:SEGMents:FREQuency:MAXimum
value: ResultData = driver.wlanMeas.multiEval.tsMask.segments.frequency.maximum.
↪read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.10.12.7 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:FREQuency:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:FREQuency:MINimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:FREQuency:MINimum
```

class MinimumCls

Minimum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tol_Seg_1: enums.ResultStatus2: No parameter help available
- Out_Of_Tol_Seg_2: enums.ResultStatus2: No parameter help available
- Margin_Xvals_Seg_1: List[enums.ResultStatus2]: No parameter help available
- Margin_Xvals_Seg_2: List[enums.ResultStatus2]: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tol_Seg_1: float: No parameter help available
- Out_Of_Tol_Seg_2: float: No parameter help available
- Margin_Xvals_Seg_1: List[float]: No parameter help available
- Margin_Xvals_Seg_2: List[float]: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>
↪:MEValuation:TSMask:SEGMents:FREQuency:MINimum
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.segments.frequency.
↪minimum.calculate()
```

No command help available

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>
↪:MEValuation:TSMask:SEGMents:FREQuency:MINimum
value: ResultData = driver.wlanMeas.multiEval.tsMask.segments.frequency.minimum.
↪fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>
↪:MEValuation:TSMask:SEGMents:FREQuency:MINimum
value: ResultData = driver.wlanMeas.multiEval.tsMask.segments.frequency.minimum.
↪read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.10.12.8 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:MAXimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:MAXimum
```

class MaximumCls

Maximum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tol_Seg_1: enums.ResultStatus2: No parameter help available
- Out_Of_Tol_Seg_2: enums.ResultStatus2: No parameter help available
- Margin_Seg_1: List[enums.ResultStatus2]: No parameter help available
- Margin_Seg_2: List[enums.ResultStatus2]: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tol_Seg_1: float: No parameter help available
- Out_Of_Tol_Seg_2: float: No parameter help available
- Margin_Seg_1: List[float]: No parameter help available
- Margin_Seg_2: List[float]: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:MAXimum
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.segments.maximum.
↪calculate()
```

No command help available

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:MAXimum
value: ResultData = driver.wlanMeas.multiEval.tsMask.segments.maximum.fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:MAXimum
value: ResultData = driver.wlanMeas.multiEval.tsMask.segments.maximum.read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.10.12.9 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:MINimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:MINimum
```

class MinimumCls

Minimum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tol_Seg_1: enums.ResultStatus2: No parameter help available
- Out_Of_Tol_Seg_2: enums.ResultStatus2: No parameter help available
- Margin_Seg_1: List[enums.ResultStatus2]: No parameter help available
- Margin_Seg_2: List[enums.ResultStatus2]: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tol_Seg_1: float: No parameter help available
- Out_Of_Tol_Seg_2: float: No parameter help available
- Margin_Seg_1: List[float]: No parameter help available
- Margin_Seg_2: List[float]: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:MINimum
value: CalculateStruct = driver.wlanMeas.multiEval.tsMask.segments.minimum.
↪calculate()
```

No command help available

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:MINimum
value: ResultData = driver.wlanMeas.multiEval.tsMask.segments.minimum.fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:MINimum
value: ResultData = driver.wlanMeas.multiEval.tsMask.segments.minimum.read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.4.1.11 UtError<UtError>

RepCap Settings

```
# Range: Nr1 .. Nr8
rc = driver.wlanMeas.multiEval.utError.repcap_utError_get()
driver.wlanMeas.multiEval.utError.repcap_utError_set(repcap.UtError.Nr1)
```

class UtErrorCls

UtError commands group definition. 27 total commands, 6 Subgroups, 0 group commands Repeated Capability: UtError, default value after init: UtError.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.utError.clone()
```

Subgroups

6.4.1.11.1 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:AVERage
CALCulate:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*utError=UtError.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:AVERage
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.utError.average.
↪calculate(utError = repcap.UtError.Default)
```

Return the values of the unused tone error traces according to standard 802.11ax and be. The results of the current, average, minimum and maximum traces can be retrieved. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param utError
optional repeated capability selector. Default value: Nr1 (settable in the interface 'UtError')

return
ute_yvals: Comma-separated list of unused tone error results, one value per each 26-tone RU. The total number of RUs depends on the bandwidth, see table below.

fetch(*utError=UtError.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:AVERage
value: List[float] = driver.wlanMeas.multiEval.utError.average.fetch(utError =␣
↪repcap.UtError.Default)
```

Return the values of the unused tone error traces according to standard 802.11ax and be. The results of the current, average, minimum and maximum traces can be retrieved. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param utError
optional repeated capability selector. Default value: Nr1 (settable in the interface 'UtError')

return
ute_yvals: Comma-separated list of unused tone error results, one value per each 26-tone RU. The total number of RUs depends on the bandwidth, see table below.

read(*utError=UtError.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:AVERage
value: List[float] = driver.wlanMeas.multiEval.utError.average.read(utError =␣
↪repcap.UtError.Default)
```

Return the values of the unused tone error traces according to standard 802.11ax and be. The results of the current, average, minimum and maximum traces can be retrieved. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param utError
optional repeated capability selector. Default value: Nr1 (settable in the interface 'UtError')

return
ute_yvals: Comma-separated list of unused tone error results, one value per each 26-tone RU. The total number of RUs depends on the bandwidth, see table below.

6.4.1.11.2 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:CURRent
CALCulate:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:CURRent
```

class CurrentCls

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*utError=UtError.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:CURRent
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.utError.current.
↪calculate(utError = repcap.UtError.Default)
```

Return the values of the unused tone error traces according to standard 802.11ax and be. The results of the current, average, minimum and maximum traces can be retrieved. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param utError
optional repeated capability selector. Default value: Nr1 (settable in the interface 'UtError')

return
ute_yvals: Comma-separated list of unused tone error results, one value per each 26-tone RU. The total number of RUs depends on the bandwidth, see table below.

fetch(*utError=UtError.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:CURRent
value: List[float] = driver.wlanMeas.multiEval.utError.current.fetch(utError =␣
↪repcap.UtError.Default)
```

Return the values of the unused tone error traces according to standard 802.11ax and be. The results of the current, average, minimum and maximum traces can be retrieved. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param utError
optional repeated capability selector. Default value: Nr1 (settable in the interface 'UtError')

return
ute_yvals: Comma-separated list of unused tone error results, one value per each 26-tone RU. The total number of RUs depends on the bandwidth, see table below.

read(*utError=UtError.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:CURRent
value: List[float] = driver.wlanMeas.multiEval.utError.current.read(utError =␣
↪repcap.UtError.Default)
```

Return the values of the unused tone error traces according to standard 802.11ax and be. The results of the current, average, minimum and maximum traces can be retrieved. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param utError
optional repeated capability selector. Default value: Nr1 (settable in the interface 'UtError')

return
ute_yvals: Comma-separated list of unused tone error results, one value per each 26-tone RU. The total number of RUs depends on the bandwidth, see table below.

6.4.1.11.3 Limit

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:LIMit
FETCh:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:LIMit
CALCulate:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:LIMit
```

class LimitCls

Limit commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*utError=UtError.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:LIMit
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.utError.limit.
↪calculate(utError = repcap.UtError.Default)
```

Displays unused tone error limit line. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param utError
optional repeated capability selector. Default value: Nr1 (settable in the interface 'UtError')

return
ute_limit_line: Comma-separated list of unused tone error limits, one value per 26-tone RU (from left to right)

fetch(*utError=UtError.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:LIMit
value: List[float] = driver.wlanMeas.multiEval.utError.limit.fetch(utError =␣
↪repcap.UtError.Default)
```

Displays unused tone error limit line. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param utError
optional repeated capability selector. Default value: Nr1 (settable in the interface 'UtError')

return
ute_limit_line: Comma-separated list of unused tone error limits, one value per 26-tone RU (from left to right)

read(*utError=UtError.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:LIMit
value: List[float] = driver.wlanMeas.multiEval.utError.limit.read(utError =␣
↪repcap.UtError.Default)
```

Displays unused tone error limit line. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param utError
optional repeated capability selector. Default value: Nr1 (settable in the interface 'UtError')

return
ute_limit_line: Comma-separated list of unused tone error limits, one value per 26-tone RU (from left to right)

6.4.1.11.4 Margin

class MarginCls

Margin commands group definition. 12 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.multiEval.utError.margin.clone()
```

Subgroups

6.4.1.11.4.1 Average

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MARGin:AVERage
FETCh:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MARGin:AVERage
CALCulate:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MARGin:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*utError=UtError.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>
↪:MARGin:AVERage
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.utError.margin.
↪average.calculate(utError = repcap.UtError.Default)
```

Returns the margin values of the unused tone error measurement for the current, average, minimum and maximum traces. A positive margin indicates a violation of the unused tone error limit line. The respective trace value is located above the upper limit line. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param utError
optional repeated capability selector. Default value: Nr1 (settable in the interface 'UtError')

return
ute_margin: Comma-separated list of margins to the unused tone error limits, one value per each 26-tone RU. The total number of RUs depends on the bandwidth, see table below.

fetch(*utError=UtError.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MARGin:AVERage
value: List[float] = driver.wlanMeas.multiEval.utError.margin.average.
↪fetch(utError = repcap.UtError.Default)
```

Returns the margin values of the unused tone error measurement for the current, average, minimum and maximum traces. A positive margin indicates a violation of the unused tone error limit line. The respective trace value is located above the upper limit line. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param utError
optional repeated capability selector. Default value: Nr1 (settable in the interface 'UtError')

return
ute_margin: Comma-separated list of margins to the unused tone error limits, one value per each 26-tone RU. The total number of RUs depends on the bandwidth, see table below.

read(*utError=UtError.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MARGin:AVERage
value: List[float] = driver.wlanMeas.multiEval.utError.margin.average.
↪read(utError = repcap.UtError.Default)
```

Returns the margin values of the unused tone error measurement for the current, average, minimum and maximum traces. A positive margin indicates a violation of the unused tone error limit line. The respective trace value is located above the upper limit line. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param utError
optional repeated capability selector. Default value: Nr1 (settable in the interface 'UtError')

return
ute_margin: Comma-separated list of margins to the unused tone error limits, one value per each 26-tone RU. The total number of RUs depends on the bandwidth, see table below.

6.4.1.11.4.2 Current

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MARGin:CURRent
FETCh:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MARGin:CURRent
CALCulate:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MARGin:CURRent
```

class CurrentCls

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*utError=UtError.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>
↪:MARGin:CURRent
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.utError.margin.
↪current.calculate(utError = repcap.UtError.Default)
```

Returns the margin values of the unused tone error measurement for the current, average, minimum and maximum traces. A positive margin indicates a violation of the unused tone error limit line. The respective trace value is located above the upper limit line. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param utError
optional repeated capability selector. Default value: Nr1 (settable in the interface 'UtError')

return
ute_margin: Comma-separated list of margins to the unused tone error limits, one value per each 26-tone RU. The total number of RUs depends on the bandwidth, see table below.

fetch(*utError=UtError.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MARGin:CURRent
value: List[float] = driver.wlanMeas.multiEval.utError.margin.current.
↪fetch(utError = repcap.UtError.Default)
```

Returns the margin values of the unused tone error measurement for the current, average, minimum and maximum traces. A positive margin indicates a violation of the unused tone error limit line. The respective trace value is located above the upper limit line. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param utError
optional repeated capability selector. Default value: Nr1 (settable in the interface 'UtError')

return
ute_margin: Comma-separated list of margins to the unused tone error limits, one value per each 26-tone RU. The total number of RUs depends on the bandwidth, see table below.

read(*utError=UtError.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MARGin:CURRent
value: List[float] = driver.wlanMeas.multiEval.utError.margin.current.
↪read(utError = repcap.UtError.Default)
```

Returns the margin values of the unused tone error measurement for the current, average, minimum and maximum traces. A positive margin indicates a violation of the unused tone error limit line. The respective trace value is located above the upper limit line. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param utError
optional repeated capability selector. Default value: Nr1 (settable in the interface 'UtError')

return
ute_margin: Comma-separated list of margins to the unused tone error limits, one value per each 26-tone RU. The total number of RUs depends on the bandwidth, see table below.

6.4.1.11.4.3 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MARGin:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MARGin:MAXimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MARGin:MAXimum
```

class MaximumCls

Maximum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*utError=UtError.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>
↪:MARGin:MAXimum
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.utError.margin.
↪maximum.calculate(utError = repcap.UtError.Default)
```

Returns the margin values of the unused tone error measurement for the current, average, minimum and maximum traces. A positive margin indicates a violation of the unused tone error limit line. The respective trace value is located above the upper limit line. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param utError
optional repeated capability selector. Default value: Nr1 (settable in the interface 'UtError')

return
ute_margin: Comma-separated list of margins to the unused tone error limits, one value per each 26-tone RU. The total number of RUs depends on the bandwidth, see table below.

fetch(*utError=UtError.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MARGin:MAXimum
value: List[float] = driver.wlanMeas.multiEval.utError.margin.maximum.
↪fetch(utError = repcap.UtError.Default)
```

Returns the margin values of the unused tone error measurement for the current, average, minimum and maximum traces. A positive margin indicates a violation of the unused tone error limit line. The respective trace value is located above the upper limit line. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param utError
optional repeated capability selector. Default value: Nr1 (settable in the interface 'UtError')

return
ute_margin: Comma-separated list of margins to the unused tone error limits, one value per each 26-tone RU. The total number of RUs depends on the bandwidth, see table below.

read(*utError=UtError.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MARGin:MAXimum
value: List[float] = driver.wlanMeas.multiEval.utError.margin.maximum.
↪read(utError = repcap.UtError.Default)
```

Returns the margin values of the unused tone error measurement for the current, average, minimum and maximum traces. A positive margin indicates a violation of the unused tone error limit line. The respective trace value is located above the upper limit line. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param utError
optional repeated capability selector. Default value: Nr1 (settable in the interface 'UtError')

return
ute_margin: Comma-separated list of margins to the unused tone error limits, one value per each 26-tone RU. The total number of RUs depends on the bandwidth, see table below.

6.4.1.11.4.4 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MARGin:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MARGin:MINimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MARGin:MINimum
```

class MinimumCls

Minimum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*utError=UtError.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>
↪:MARGin:MINimum
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.utError.margin.
↪minimum.calculate(utError = repcap.UtError.Default)
```

Returns the margin values of the unused tone error measurement for the current, average, minimum and maximum traces. A positive margin indicates a violation of the unused tone error limit line. The respective trace value is located above the upper limit line. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param utError
optional repeated capability selector. Default value: Nr1 (settable in the interface 'UtError')

return
ute_margin: Comma-separated list of margins to the unused tone error limits, one

value per each 26-tone RU. The total number of RUs depends on the bandwidth, see table below.

fetch(*utError=UtError.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MARGin:MINimum
value: List[float] = driver.wlanMeas.multiEval.utError.margin.minimum.
↪fetch(utError = repcap.UtError.Default)
```

Returns the margin values of the unused tone error measurement for the current, average, minimum and maximum traces. A positive margin indicates a violation of the unused tone error limit line. The respective trace value is located above the upper limit line. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param utError
optional repeated capability selector. Default value: Nr1 (settable in the interface 'UtError')

return
ute_margin: Comma-separated list of margins to the unused tone error limits, one value per each 26-tone RU. The total number of RUs depends on the bandwidth, see table below.

read(*utError=UtError.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MARGin:MINimum
value: List[float] = driver.wlanMeas.multiEval.utError.margin.minimum.
↪read(utError = repcap.UtError.Default)
```

Returns the margin values of the unused tone error measurement for the current, average, minimum and maximum traces. A positive margin indicates a violation of the unused tone error limit line. The respective trace value is located above the upper limit line. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param utError
optional repeated capability selector. Default value: Nr1 (settable in the interface 'UtError')

return
ute_margin: Comma-separated list of margins to the unused tone error limits, one value per each 26-tone RU. The total number of RUs depends on the bandwidth, see table below.

6.4.1.11.5 Maximum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MAXimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MAXimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MAXimum
```

class MaximumCls

Maximum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*utError=UtError.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MAXimum
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.utError.maximum.
↪calculate(utError = repcap.UtError.Default)
```

Return the values of the unused tone error traces according to standard 802.11ax and be. The results of the current, average, minimum and maximum traces can be retrieved. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param utError
optional repeated capability selector. Default value: Nr1 (settable in the interface 'UtError')

return
ute_yvals: Comma-separated list of unused tone error results, one value per each 26-tone RU. The total number of RUs depends on the bandwidth, see table below.

fetch(*utError=UtError.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MAXimum
value: List[float] = driver.wlanMeas.multiEval.utError.maximum.fetch(utError =␣
↪repcap.UtError.Default)
```

Return the values of the unused tone error traces according to standard 802.11ax and be. The results of the current, average, minimum and maximum traces can be retrieved. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param utError
optional repeated capability selector. Default value: Nr1 (settable in the interface 'UtError')

return
ute_yvals: Comma-separated list of unused tone error results, one value per each 26-tone RU. The total number of RUs depends on the bandwidth, see table below.

read(*utError=UtError.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MAXimum
value: List[float] = driver.wlanMeas.multiEval.utError.maximum.read(utError =␣
↪repcap.UtError.Default)
```

Return the values of the unused tone error traces according to standard 802.11ax and be. The results of the current, average, minimum and maximum traces can be retrieved. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param utError
optional repeated capability selector. Default value: Nr1 (settable in the interface 'UtError')

return
ute_yvals: Comma-separated list of unused tone error results, one value per each 26-tone RU. The total number of RUs depends on the bandwidth, see table below.

6.4.1.11.6 Minimum

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MINimum
FETCh:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MINimum
CALCulate:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MINimum
```

class MinimumCls

Minimum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

calculate(*utError=UtError.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MINimum
value: List[enums.ResultStatus2] = driver.wlanMeas.multiEval.utError.minimum.
↪calculate(utError = repcap.UtError.Default)
```

Return the values of the unused tone error traces according to standard 802.11ax and be. The results of the current, average, minimum and maximum traces can be retrieved. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param utError
optional repeated capability selector. Default value: Nr1 (settable in the interface 'UtError')

return
ute_yvals: Comma-separated list of unused tone error results, one value per each 26-tone RU. The total number of RUs depends on the bandwidth, see table below.

fetch(*utError=UtError.Default*) → List[float]

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MINimum
value: List[float] = driver.wlanMeas.multiEval.utError.minimum.fetch(utError =␣
↪repcap.UtError.Default)
```

Return the values of the unused tone error traces according to standard 802.11ax and be. The results of the current, average, minimum and maximum traces can be retrieved. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param utError
optional repeated capability selector. Default value: Nr1 (settable in the interface 'UtError')

return
ute_yvals: Comma-separated list of unused tone error results, one value per each 26-tone RU. The total number of RUs depends on the bandwidth, see table below.

read(*utError=UtError.Default*) → List[float]

```
# SCPI: READ:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MINimum
value: List[float] = driver.wlanMeas.multiEval.utError.minimum.read(utError =␣
↪repcap.UtError.Default)
```

Return the values of the unused tone error traces according to standard 802.11ax and be. The results of the current, average, minimum and maximum traces can be retrieved. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Suppressed linked return values: reliability

param utError
optional repeated capability selector. Default value: Nr1 (settable in the interface 'UtError')

return
ute_yvals: Comma-separated list of unused tone error results, one value per each 26-tone RU. The total number of RUs depends on the bandwidth, see table below.

6.4.2 Tmode

SCPI Command :

```
ABORt:WLAN:MEASurement<Instance>:TMODe
```

class TmodeCls

Tmode commands group definition. 5 total commands, 2 Subgroups, 1 group commands

abort(*opc_timeout_ms: int = -1*) → None

```
# SCPI: ABORt:WLAN:MEASurement<Instance>:TMODe
driver.wlanMeas.tmode.abort()
```

No command help available

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.tmode.clone()
```

Subgroups

6.4.2.1 Antenna<Antenna>

RepCap Settings

```
# Range: Nr1 .. Nr8
rc = driver.wlanMeas.tmode.antenna.repcap_antenna_get()
driver.wlanMeas.tmode.antenna.repcap_antenna_set(repcap.Antenna.Nr1)
```

SCPI Commands :

```
READ:WLAN:MEASurement<Instance>:TMODe:ANTenna<Antennas>
FETCh:WLAN:MEASurement<Instance>:TMODe:ANTenna<Antennas>
INITiate:WLAN:MEASurement<Instance>:TMODe:ANTenna<Antennas>
```

class AntennaCls

Antenna commands group definition. 3 total commands, 0 Subgroups, 3 group commands Repeated Capability: Antenna, default value after init: Antenna.Nr1

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Decode_Status: enums.DecodeStatus: No parameter help available
- Mcs: int: No parameter help available
- Power: float: No parameter help available
- Pilot_Evm: float: No parameter help available

fetch(*antenna=Antenna.Default*) → ResultData

```
# SCPI: FETCh:WLAN:MEASurement<Instance>:TMODe:ANTenna<Antennas>
value: ResultData = driver.wlanMeas.tmode.antenna.fetch(antenna = repcap.
↪Antenna.Default)
```

No command help available

param antenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Antenna')

return
structure: for return value, see the help for ResultData structure arguments.

initiate(*antenna=Antenna.Default*, *opc_timeout_ms: int = -1*) → None

```
# SCPI: INITiate:WLAN:MEASurement<Instance>:TMODe:ANTenna<Antennas>
driver.wlanMeas.tmode.antenna.initiate(antenna = repcap.Antenna.Default)
```

No command help available

param antenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Antenna')

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

read(*antenna=Antenna.Default*) → ResultData

```
# SCPI: READ:WLAN:MEASurement<Instance>:TMODe:ANTenna<Antennas>
value: ResultData = driver.wlanMeas.tmode.antenna.read(antenna = repcap.Antenna.
↪Default)
```

No command help available

param antenna
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Antenna')

return
structure: for return value, see the help for ResultData structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.wlanMeas.tmode.antenna.clone()
```

6.4.2.2 Data

SCPI Command :

```
CLEar:WLAN:MEASurement<instance>:TMODe:DATA
```

class DataCls

Data commands group definition. 1 total commands, 0 Subgroups, 1 group commands

clear() → None

```
# SCPI: CLEar:WLAN:MEASurement<instance>:TMODe:DATA
driver.wlanMeas.tmode.data.clear()
```

No command help available

clear_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: CLEar:WLAN:MEASurement<instance>:TMODe:DATA
driver.wlanMeas.tmode.data.clear_with_opc()
```

No command help available

Same as clear, but waits for the operation to complete before continuing further. Use the RsCMPX_WlanMeas.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

CHAPTER

SEVEN

RSCMPX_WLANMEAS UTILITIES

class Utilities

Common utility class. Utility functions common for all types of drivers.

Access snippet: `utils = RsCMPX_WlanMeas.utilities`

property logger: *ScpiLogger*

Scpi Logger interface, see *here*

Access snippet: `logger = RsCMPX_WlanMeas.utilities.logger`

property driver_version: str

Returns the instrument driver version.

property idn_string: str

Returns instrument's identification string - the response on the SCPI command *IDN?

property manufacturer: str

Returns manufacturer of the instrument.

property full_instrument_model_name: str

Returns the current instrument's full name e.g. 'FSW26'.

property instrument_model_name: str

Returns the current instrument's family name e.g. 'FSW'.

property supported_models: List[str]

Returns a list of the instrument models supported by this instrument driver.

property instrument_firmware_version: str

Returns instrument's firmware version.

property instrument_serial_number: str

Returns instrument's serial_number.

query_opc(*timeout: int = 0*) → int

SCPI command: *OPC? Queries the instrument's OPC bit and hence it waits until the instrument reports operation complete. If you define timeout > 0, the VISA timeout is set to that value just for this method call.

property instrument_status_checking: bool

Sets / returns Instrument Status Checking. When True (default is True), all the driver methods and properties are sending "SYSTem:ERRor?" at the end to immediately react on error that might have occurred. We recommend to keep the state checking ON all the time. Switch it OFF only in rare cases when you require maximum speed. The default state after initializing the session is ON.

property encoding: str

Returns string<=>bytes encoding of the session.

property opc_query_after_write: bool

Sets / returns Instrument *OPC? query sending after each command write. When True, (default is False) the driver sends *OPC? every time a write command is performed. Use this if you want to make sure your sequence is performed command-after-command.

property bin_float_numbers_format: BinFloatFormat

Sets / returns format of float numbers when transferred as binary data.

property bin_int_numbers_format: BinIntFormat

Sets / returns format of integer numbers when transferred as binary data.

clear_status() → None

Clears instrument's status system, the session's I/O buffers and the instrument's error queue.

query_all_errors() → List[str]

Queries and clears all the errors from the instrument's error queue. The method returns list of strings as error messages. If no error is detected, the return value is None. The process is: querying 'SYSTem:ERRor?' in a loop until the error queue is empty. If you want to include the error codes, call the query_all_errors_with_codes()

query_all_errors_with_codes() → List[Tuple[int, str]]

Queries and clears all the errors from the instrument's error queue. The method returns list of tuples (code: int, message: str). If no error is detected, the return value is None. The process is: querying 'SYSTem:ERRor?' in a loop until the error queue is empty.

property instrument_options: List[str]

Returns all the instrument options. The options are sorted in the ascending order starting with K-options and continuing with B-options.

reset() → None

SCPI command: *RST Sends *RST command + calls the clear_status().

default_instrument_setup() → None

Custom steps performed at the init and at the reset().

self_test(*timeout: int = None*) → Tuple[int, str]

SCPI command: *TST? Performs instrument's self-test. Returns tuple (code:int, message: str). Code 0 means the self-test passed. You can define the custom timeout in milliseconds. If you do not define it, the default selftest timeout is used (usually 60 secs).

is_connection_active() → bool

Returns true, if the VISA connection is active and the communication with the instrument still works.

reconnect(*force_close: bool = False*) → bool

If the connection is not active, the method tries to reconnect to the device If the connection is active, and force_close is False, the method does nothing. If the connection is active, and force_close is True, the method closes, and opens the session again. Returns True, if the reconnection has been performed.

property resource_name: int

Returns the resource name used in the constructor

property opc_timeout: int

Sets / returns timeout in milliseconds for all the operations that use OPC synchronization.

property visa_timeout: int

Sets / returns visa IO timeout in milliseconds.

property data_chunk_size: int

Sets / returns the maximum size of one block transferred during write/read operations

property visa_manufacturer: int

Returns the manufacturer of the current VISA session.

process_all_commands() → None

SCPI command: *WAI Stops further commands processing until all commands sent before *WAI have been executed.

write_str(*cmd: str*) → None

Writes the command to the instrument.

write(*cmd: str*) → None

This method is an alias to the write_str(). Writes the command to the instrument as string.

write_int(*cmd: str*, *param: int*) → None

Writes the command to the instrument followed by the integer parameter: e.g.: cmd = 'SELECT:INPUT' param = '2', result command = 'SELECT:INPUT 2'

write_int_with_opc(*cmd: str*, *param: int*, *timeout: int = None*) → None

Writes the command with OPC to the instrument followed by the integer parameter: e.g.: cmd = 'SELECT:INPUT' param = '2', result command = 'SELECT:INPUT 2' If you do not provide timeout, the method uses current opc_timeout.

write_float(*cmd: str*, *param: float*) → None

Writes the command to the instrument followed by the boolean parameter: e.g.: cmd = 'CENTER:FREQ' param = '10E6', result command = 'CENTER:FREQ 10E6'

write_float_with_opc(*cmd: str*, *param: float*, *timeout: int = None*) → None

Writes the command with OPC to the instrument followed by the boolean parameter: e.g.: cmd = 'CENTER:FREQ' param = '10E6', result command = 'CENTER:FREQ 10E6' If you do not provide timeout, the method uses current opc_timeout.

write_bool(*cmd: str*, *param: bool*) → None

Writes the command to the instrument followed by the boolean parameter: e.g.: cmd = 'OUTPUT' param = 'True', result command = 'OUTPUT ON'

write_bool_with_opc(*cmd: str*, *param: bool*, *timeout: int = None*) → None

Writes the command with OPC to the instrument followed by the boolean parameter: e.g.: cmd = 'OUTPUT' param = 'True', result command = 'OUTPUT ON' If you do not provide timeout, the method uses current opc_timeout.

query_str(*query: str*) → str

Sends the query to the instrument and returns the response as string. The response is trimmed of any trailing LF characters and has no length limit.

query(*query: str*) → str

This method is an alias to the query_str(). Sends the query to the instrument and returns the response as string. The response is trimmed of any trailing LF characters and has no length limit.

query_bool(*query: str*) → bool

Sends the query to the instrument and returns the response as boolean.

query_int(*query: str*) → int

Sends the query to the instrument and returns the response as integer.

query_float(*query: str*) → float

Sends the query to the instrument and returns the response as float.

write_str_with_opc(*cmd: str*, *timeout: int = None*) → None

Writes the opc-synced command to the instrument. If you do not provide timeout, the method uses current opc_timeout.

write_with_opc(*cmd: str*, *timeout: int = None*) → None

This method is an alias to the write_str_with_opc(). Writes the opc-synced command to the instrument. If you do not provide timeout, the method uses current opc_timeout.

query_str_with_opc(*query: str*, *timeout: int = None*) → str

Sends the opc-synced query to the instrument and returns the response as string. The response is trimmed of any trailing LF characters and has no length limit. If you do not provide timeout, the method uses current opc_timeout.

query_with_opc(*query: str*, *timeout: int = None*) → str

This method is an alias to the query_str_with_opc(). Sends the opc-synced query to the instrument and returns the response as string. The response is trimmed of any trailing LF characters and has no length limit. If you do not provide timeout, the method uses current opc_timeout.

query_bool_with_opc(*query: str*, *timeout: int = None*) → bool

Sends the opc-synced query to the instrument and returns the response as boolean. If you do not provide timeout, the method uses current opc_timeout.

query_int_with_opc(*query: str*, *timeout: int = None*) → int

Sends the opc-synced query to the instrument and returns the response as integer. If you do not provide timeout, the method uses current opc_timeout.

query_float_with_opc(*query: str*, *timeout: int = None*) → float

Sends the opc-synced query to the instrument and returns the response as float. If you do not provide timeout, the method uses current opc_timeout.

write_bin_block(*cmd: str*, *payload: bytes*) → None

Writes all the payload as binary data block to the instrument. The binary data header is added at the beginning of the transmission automatically, do not include it in the payload!!!

query_bin_block(*query: str*) → bytes

Queries binary data block to bytes. Throws an exception if the returned data was not a binary data. Returns data:bytes

query_bin_block_with_opc(*query: str*, *timeout: int = None*) → bytes

Sends a OPC-synced query and returns binary data block to bytes. If you do not provide timeout, the method uses current opc_timeout.

query_bin_or_ascii_float_list(*query: str*) → List[float]

Queries a list of floating-point numbers that can be returned in ASCII format or in binary format. - For ASCII format, the list numbers are decoded as comma-separated values. - For Binary Format, the numbers are decoded based on the property BinFloatFormat, usually float 32-bit (FORM REAL,32).

query_bin_or_ascii_float_list_with_opc(*query: str*, *timeout: int = None*) → List[float]

Sends a OPC-synced query and reads a list of floating-point numbers that can be returned in ASCII format or in binary format. - For ASCII format, the list numbers are decoded as comma-separated values. - For Binary Format, the numbers are decoded based on the property BinFloatFormat, usually float 32-bit (FORM REAL,32). If you do not provide timeout, the method uses current opc_timeout.

query_bin_or_ascii_int_list(*query: str*) → List[int]

Queries a list of floating-point numbers that can be returned in ASCII format or in binary format. - For ASCII format, the list numbers are decoded as comma-separated values. - For Binary Format, the numbers are decoded based on the property BinFloatFormat, usually float 32-bit (FORM REAL,32).

query_bin_or_ascii_int_list_with_opc(*query: str*, *timeout: int = None*) → List[int]

Sends a OPC-synced query and reads a list of floating-point numbers that can be returned in ASCII format or in binary format. - For ASCII format, the list numbers are decoded as comma-separated values. - For Binary Format, the numbers are decoded based on the property BinFloatFormat, usually float 32-bit (FORM REAL,32). If you do not provide timeout, the method uses current opc_timeout.

query_bin_block_to_file(*query: str*, *file_path: str*, *append: bool = False*) → None

Queries binary data block to the provided file. If append is False, any existing file content is discarded. If append is True, the new content is added to the end of the existing file, or if the file does not exit, it is created. Throws an exception if the returned data was not a binary data. Example for transferring a file from Instrument -> PC: query = f'MMEM:DATA? '{INSTR_FILE_PATH}'". Alternatively, use the dedicated methods for this purpose:

- `send_file_from_pc_to_instrument()`
- `read_file_from_instrument_to_pc()`

query_bin_block_to_file_with_opc(*query: str*, *file_path: str*, *append: bool = False*, *timeout: int = None*) → None

Sends a OPC-synced query and writes the returned data to the provided file. If append is False, any existing file content is discarded. If append is True, the new content is added to the end of the existing file, or if the file does not exit, it is created. Throws an exception if the returned data was not a binary data.

write_bin_block_from_file(*cmd: str*, *file_path: str*) → None

Writes data from the file as binary data block to the instrument using the provided command. Example for transferring a file from PC -> Instrument: cmd = f'MMEM:DATA '{INSTR_FILE_PATH}',". Alternatively, use the dedicated methods for this purpose:

- `send_file_from_pc_to_instrument()`
- `read_file_from_instrument_to_pc()`

send_file_from_pc_to_instrument(*source_pc_file: str*, *target_instr_file: str*) → None

SCPI Command: MMEM:DATA

Sends file from PC to the instrument

read_file_from_instrument_to_pc(*source_instr_file: str*, *target_pc_file: str*, *append_to_pc_file: bool = False*) → None

SCPI Command: MMEM:DATA?

Reads file from instrument to the PC.

Set the `append_to_pc_file` to True if you want to append the read content to the end of the existing PC file

get_last_sent_cmd() → str

Returns the last commands sent to the instrument. Only works in simulation mode

go_to_local() → None

Puts the instrument into local state.

go_to_remote() → None

Puts the instrument into remote state.

get_lock() → RLock

Returns the thread lock for the current session.

By default:

- If you create standard new RsCMPX_WlanMeas instance with new VISA session, the session gets a new thread lock. You can assign it to other RsCMPX_WlanMeas sessions in order to share one physical instrument with a multi-thread access.
- If you create new RsCMPX_WlanMeas from an existing session, the thread lock is shared automatically making both instances multi-thread safe.

You can always assign new thread lock by calling `driver.utilities.assign_lock()`

assign_lock(*lock: RLock*) → None

Assigns the provided thread lock.

clear_lock()

Clears the existing thread lock, making the current session thread-independent from others that might share the current thread lock.

sync_from(*source: Utilities*) → None

Synchronises these Utils with the source.

CHAPTER

EIGHT

RSCMPX_WLANMEAS LOGGER

Check the usage in the Getting Started chapter *here*.

class ScpiLogger

Base class for SCPI logging

mode

Sets the logging ON or OFF. Additionally, you can set the logging ON only for errors. Possible values:

- LoggingMode.Off - logging is switched OFF
- LoggingMode.On - logging is switched ON
- LoggingMode.Errors - logging is switched ON, but only for error entries
- LoggingMode.Default - sets the logging to default - the value you have set with logger.default_mode

default_mode

Sets / returns the default logging mode. You can recall the default mode by calling the logger.mode = LoggingMode.Default.

Data Type
LoggingMode

device_name: str

Use this property to change the resource name in the log from the default Resource Name (e.g. TCPIP::192.168.2.101::INSTR) to another name e.g. 'MySigGen1'.

set_logging_target(*target*, *console_log: bool = None*, *udp_log: bool = None*) → None

Sets logging target - the target must implement write() and flush(). You can optionally set the console and UDP logging ON or OFF. This method switches the logging target global OFF.

get_logging_target()

Based on the global_mode, it returns the logging target: either the local or the global one.

set_logging_target_global(*console_log: bool = None*, *udp_log: bool = None*) → None

Sets logging target to global. The global target must be defined. You can optionally set the console and UDP logging ON or OFF.

log_to_console

Returns logging to console status.

log_to_udp

Returns logging to UDP status.

log_to_console_and_udp

Returns true, if both logging to UDP and console in are True.

info_raw(*log_entry: str*, *add_new_line: bool = True*) → None

Method for logging the raw string without any formatting.

info(*start_time: datetime*, *end_time: datetime*, *log_string_info: str*, *log_string: str*) → None

Method for logging one info entry. For binary log_string, use the info_bin()

error(*start_time: datetime*, *end_time: datetime*, *log_string_info: str*, *log_string: str*) → None

Method for logging one error entry.

set_relative_timestamp(*timestamp: datetime*) → None

If set, the further timestamps will be relative to the entered time.

set_relative_timestamp_now() → None

Sets the relative timestamp to the current time.

get_relative_timestamp() → datetime

Based on the global_mode, it returns the relative timestamp: either the local or the global one.

clear_relative_timestamp() → None

Clears the reference time, and the further logging continues with absolute times.

flush() → None

Flush all the entries.

log_status_check_ok

Sets / returns the current status of status checking OK. If True (default), the log contains logging of the status checking 'Status check: OK'. If False, the 'Status check: OK' is skipped - the log is more compact. Errors will still be logged.

clear_cached_entries() → None

Clears potential cached log entries. Cached log entries are generated when the Logging is ON, but no target has been defined yet.

set_format_string(*value: str*, *line_divider: str = '\n'*) → None

Sets new format string and line divider. If you just want to set the line divider, set the format string value=None The original format string is: `PAD_LEFT12(%START_TIME%) PAD_LEFT25(%DEVICE_NAME%) PAD_LEFT12(%DURATION%) %LOG_STRING_INFO%: %LOG_STRING%`

restore_format_string() → None

Restores the original format string and the line divider to LF

abbreviated_max_len_ascii: int

Defines the maximum length of one ASCII log entry. Default value is 200 characters.

abbreviated_max_len_bin: int

Defines the maximum length of one Binary log entry. Default value is 2048 bytes.

abbreviated_max_len_list: int

Defines the maximum length of one list entry. Default value is 100 elements.

bin_line_block_size: int

Defines number of bytes to display in one line. Default value is 16 bytes.

udp_port

Returns udp logging port.

target_auto_flushing

Returns status of the auto-flushing for the logging target.

CHAPTER

NINE

RSCMPX_WLANMEAS EVENTS

Check the usage in the Getting Started chapter *here*.

class Events

Common Events class. Event-related methods and properties. Here you can set all the event handlers.

property before_query_handler: Callable

Returns the handler of before_query events.

Returns
current `before_query_handler`

property before_write_handler: Callable

Returns the handler of before_write events.

Returns
current `before_write_handler`

property io_events_include_data: bool

Returns the current state of the io_events_include_data See the setter for more details.

property on_read_handler: Callable

Returns the handler of on_read events.

Returns
current `on_read_handler`

property on_write_handler: Callable

Returns the handler of on_write events.

Returns
current `on_write_handler`

sync_from(*source: Events*) → None

Synchronises these Events with the source.

CHAPTER

TEN

INDEX

INDEX

A

abbreviated_max_len_ascii (*ScpiLogger attribute*), 868
abbreviated_max_len_bin (*ScpiLogger attribute*), 868
abbreviated_max_len_list (*ScpiLogger attribute*), 868
ABORt:WLAN:MEASurement<Instance>:MEValuation, 226
ABORt:WLAN:MEASurement<Instance>:TMODe, 857

B

bin_line_block_size (*ScpiLogger attribute*), 868

C

CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:ACSiso:AVERage, 313
CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:ACSiso:CURRent, 315
CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:ACSiso:MAXimum, 316
CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:ACSiso:SDEViation, 318
CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:AVERage, 320
CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:CFDistrib, 323
CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:CURRent, 333
CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:DSSS:AVERage, 336
CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:DSSS:CURRent, 338
CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:DSSS:MAXimum, 340
CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:DSSS:MINimum, 342
CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:DSSS:SDEViation, 344
CALCulate:WLAN:MEASurement<Instance>:MEValuation:MODulation:MAXimum, 355
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulatio, 358
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulatio, 360
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulatio, 363
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulatio, 365
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulatio, 382
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulatio, 367
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulatio, 370
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulatio, 373
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulatio, 376
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulatio, 379
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulatio, 384
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulatio, 387
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulatio, 389
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulatio, 390
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulatio, 392
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulatio, 418
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulatio, 394
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulatio, 397
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulatio, 399
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulatio, 402
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulatio, 404

CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulation:WLAN:SFLatness:AVERage, 407
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulation:WLAN:SFLatness:CURRent, 410
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulation:WLAN:SFLatness:MAXimum, 412
CALCulate:WLAN:MEASurement<instance>:MEValuation:MODulation:WLAN:SFLatness:SDEViation, 415
CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:AVERage, 431
CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:CURRent, 433
CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:MAXimum, 434
CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:MIMO:MEASurement, 436
CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:SDEViation, 437
CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:WLAN:MEASurement, 439
CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:WLAN:MEASurement:AVERage, 440
CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:WLAN:MEASurement:CURRent, 441
CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:WLAN:MEASurement:MAXimum, 443
CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:WLAN:MEASurement:MINimum, 444
CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:WLAN:MEASurement:SDEViation, 445
CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:WLAN:MEASurement:MIMO:AVERage, 446
CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:WLAN:MEASurement:MIMO:CURRent, 447
CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:WLAN:MEASurement:MIMO:MAXimum, 449
CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:WLAN:MEASurement:MIMO:MINimum, 450
CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:WLAN:MEASurement:MIMO:SDEViation, 451
CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:WLAN:MEASurement:MINimum, 453
CALCulate:WLAN:MEASurement<Instance>:MEValuation:PVTime:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:WLAN:MEASurement:SDEViation, 454
CALCulate:WLAN:MEASurement<instance>:MEValuation:SFLatness:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:AVERage, 513
CALCulate:WLAN:MEASurement<instance>:MEValuation:SFLatness:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:CURRent, 514
CALCulate:WLAN:MEASurement<instance>:MEValuation:SFLatness:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:MAXimum, 516
CALCulate:WLAN:MEASurement<instance>:MEValuation:SFLatness:WLAN:MEASurement:AVERage<Instance>:MEValuation:TRACe:SFL:AVERage, 518
CALCulate:WLAN:MEASurement<instance>:MEValuation:SFLatness:WLAN:MEASurement:CURRent<Instance>:MEValuation:TRACe:SFL:CURRent, 520
CALCulate:WLAN:MEASurement<instance>:MEValuation:SFLatness:WLAN:MEASurement:MAXimum<Instance>:MEValuation:TRACe:SFL:MAXimum, 522
CALCulate:WLAN:MEASurement<instance>:MEValuation:SFLatness:WLAN:MEASurement:MINimum<Instance>:MEValuation:TRACe:SFL:MINimum, 524
CALCulate:WLAN:MEASurement<instance>:MEValuation:SFLatness:WLAN:MEASurement:SDEViation<Instance>:MEValuation:TRACe:SFL:SDEViation, 531
CALCulate:WLAN:MEASurement<instance>:MEValuation:TRACe:SFL:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:WLAN:MEASurement:MIMO:SDEViation, 673
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:WLAN:MEASurement:MIMO:SDEViation:AVERage, 675
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:WLAN:MEASurement:MIMO:SDEViation:CURRent, 676
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:WLAN:MEASurement:MIMO:SDEViation:MAXimum, 678
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:WLAN:MEASurement:MIMO:SDEViation:MINimum, 680
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:WLAN:MEASurement:MINimum, 682
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:WLAN:MEASurement:SDEViation, 684
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:WLAN:MEASurement:SDEViation:AVERage, 685
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:WLAN:MEASurement:SDEViation:CURRent, 691
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:WLAN:MEASurement:SDEViation:MAXimum, 692
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:WLAN:MEASurement:SDEViation:MINimum, 693
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:WLAN:MEASurement:SDEViation:MIMO:AVERage, 695
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:WLAN:MEASurement:SDEViation:MIMO:CURRent, 697
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:WLAN:MEASurement:SDEViation:MIMO:MAXimum, 700
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:WLAN:MEASurement:SDEViation:MIMO:MINimum, 702
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:WLAN:MEASurement:SDEViation:MIMO:SDEViation, 704
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:WLAN:MEASurement:SDEViation:MINimum, 706
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:WLAN:MEASurement:SDEViation:SDEViation, 709
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:WLAN:MEASurement:SDEViation:SDEViation:AVERage, 711
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:WLAN:MEASurement:SDEViation:SDEViation:CURRent, 714
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:WLAN:MEASurement:SDEViation:SDEViation:MAXimum, 716
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:WLAN:MEASurement:SDEViation:SDEViation:MINimum, 719
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:WLAN:MEASurement:SDEViation:SDEViation:SDEViation, 721
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFL:WLAN:MEASurement:SDEViation:SDEViation:SDEViation:AVERage, 723

CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:SEGMent<m>:MEValuation:TSMask:MI
725
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:SEGMent<m>:MEValuation:TSMask:NS
728
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:SEGMent<m>:MEValuation:TSMask:OB
730
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MIMO:RXANtenna<n>:SEGMent<m>:MEValuation:TSMask:SE
732
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SEGMent<m>:MEValuation:TSMask:MI
738
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SEGMent<m>:MEValuation:TSMask:NS
739
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SEGMent<m>:MEValuation:TSMask:OB
741
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SEGMent<m>:MEValuation:TSMask:SE
743
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO:MEASurement<Instance>:MEValuation:TSMask:MI
775
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO:MEASurement<Instance>:MEValuation:TSMask:MI
777
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO:MEASurement<Instance>:MEValuation:TSMask:MI
778
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:NSS:MEASurement<Instance>:MEValuation:TSMask:NS
780
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:NSS:MEASurement<Instance>:MEValuation:TSMask:NS
781
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:NSS:MEASurement<Instance>:MEValuation:TSMask:NS
783
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:OSS:MEASurement<Instance>:MEValuation:TSMask:OB
784
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:OSS:MEASurement<Instance>:MEValuation:TSMask:OB
785
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:FREQuency:MEASurement<Instance>:MEValuation:TSMask:OB
787
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:FREQuency:MEASurement<Instance>:MEValuation:TSMask:SE
788
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:FREQuency:MEASurement<Instance>:MEValuation:TSMask:SE
790
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:FREQuency:MEASurement<Instance>:MEValuation:TSMask:SE
791
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO:MEASurement<Instance>:MEValuation:TSMask:SE
792
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO:MEASurement<Instance>:MEValuation:TSMask:SE
794
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO:MEASurement<Instance>:MEValuation:TSMask:SE
796
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO:MEASurement:FREQuency:MEASurement<Instance>:MEValuation:TSMask:SE
798
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO:MEASurement:FREQuency:MEASurement<Instance>:MEValuation:TSMask:SE
799
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO:MEASurement:FREQuency:MEASurement<Instance>:MEValuation:UTERror<n>
801
CALCulate:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO:MEASurement:FREQuency:MEASurement<Instance>:MEValuation:UTERror<n>
803
804
806
808
809
811
813
814
816
817
819
820
822
823
825
830
832
833
834
835
837
838
839
841
842
843
845
846

CALCulate:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:LIMit, 847
CALCulate:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MARGin:AVERage, 849
CALCulate:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MARGin:CURRent, 850
CALCulate:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MARGin:MAXimum, 852
CALCulate:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MARGin:MINimum, 853
CALCulate:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MAXimum, 855
CALCulate:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MINimum, 856
CLEar:WLAN:MEASurement<instance>:TMODe:DATA, 859
clear_cached_entries() (*ScpiLogger method*), 868
clear_relative_timestamp() (*ScpiLogger method*), 868
CONFigure:WLAN:MEASurement<Instance>:ISIGnal:BTYPe, 51
CONFigure:WLAN:MEASurement<Instance>:ISIGnal:BWIDth, 51
CONFigure:WLAN:MEASurement<Instance>:ISIGnal:CONStance, 51
CONFigure:WLAN:MEASurement<Instance>:ISIGnal:DSSLength, 55
CONFigure:WLAN:MEASurement<Instance>:ISIGnal:ELENgth, 51
CONFigure:WLAN:MEASurement<Instance>:ISIGnal:IQSWap, 51
CONFigure:WLAN:MEASurement<Instance>:ISIGnal:MODFilter, 51
CONFigure:WLAN:MEASurement<Instance>:ISIGnal:OFDMLength, 56
CONFigure:WLAN:MEASurement<Instance>:ISIGnal:PCLass, 51
CONFigure:WLAN:MEASurement<Instance>:ISIGnal:RMODe, 51
CONFigure:WLAN:MEASurement<Instance>:ISIGnal:STANdard, 51
CONFigure:WLAN:MEASurement<instance>:ISIGnal:TONE, 56
CONFigure:WLAN:MEASurement<instance>:ISIGnal:TONE:FILE:DATA, 57
CONFigure:WLAN:MEASurement<Instance>:MEValuation:CFOestimate, 58
CONFigure:WLAN:MEASurement<Instance>:MEValuation:COMPensation:CESTimation, 62
CONFigure:WLAN:MEASurement<Instance>:MEValuation:COMPensation:FTAPs, 63
CONFigure:WLAN:MEASurement<Instance>:MEValuation:COMPensation:NCANcel, 62
CONFigure:WLAN:MEASurement<Instance>:MEValuation:COMPensation:SKIPsymbols,

64
CONFigure:WLAN:MEASurement<Instance>:MEValuation:COMPensat 62
CONFigure:WLAN:MEASurement<Instance>:MEValuation:COMPensat 65
CONFigure:WLAN:MEASurement<Instance>:MEValuation:COMPensat 65
CONFigure:WLAN:MEASurement<Instance>:MEValuation:COMPensat 65
CONFigure:WLAN:MEASurement<Instance>:MEValuation:DEMod:FFT 67
CONFigure:WLAN:MEASurement<Instance>:MEValuation:EMEThod, 68
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MOD 69
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MOD 69
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MOD 69
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MOD 69
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MOD 69
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MOD 74
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MOD 71
CONFigure:WLAN:MEASurement<instance>:MEValuation:LIMit:MOD 71
CONFigure:WLAN:MEASurement<instance>:MEValuation:LIMit:MOD 71
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MOD 76
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MOD 71
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MOD 78
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MOD 77
CONFigure:WLAN:MEASurement<instance>:MEValuation:LIMit:MOD 79
CONFigure:WLAN:MEASurement<instance>:MEValuation:LIMit:MOD 79
CONFigure:WLAN:MEASurement<instance>:MEValuation:LIMit:MOD 79
CONFigure:WLAN:MEASurement<instance>:MEValuation:LIMit:MOD 79
CONFigure:WLAN:MEASurement<instance>:MEValuation:LIMit:MOD 83
CONFigure:WLAN:MEASurement<instance>:MEValuation:LIMit:MOD 83
CONFigure:WLAN:MEASurement<instance>:MEValuation:LIMit:MOD 83
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:MOD

```
85
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFL
77
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFL
86
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFL
86
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFL
89
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFL
86
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFL
90
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFL
90
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFL
90
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFL
90
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFL
93
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFL
93
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFL
93
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFL
93
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFL
96
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSM
96
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSM
96
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSM
99
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSM
96
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSM
100
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSM
100
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSM
100
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSM
103
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSM
104
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFL
105
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFL
106
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFL
107
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFL
108
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFL
110
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFL
111
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFL
112
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFL
112
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFL
114
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFL
112
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFL
115
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFL
116
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFL
117
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFL
119
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFL
120
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:SFL
121
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSM
122
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSM
123
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSM
123
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSM
124
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSM
125
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSM
126
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSM
127
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSM
128
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSM
129
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSM
130
CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSM
```

131

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HTOFdm:BW<bandwidth>:Y:A,

132

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HTOFdm:BW<bandwidth>:Y:B,

133

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HTOFdm:BW<bandwidth>:Y:C,

134

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HTOFdm:BW<bandwidth>:Y:D,

135

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HTOFdm:BW<bandwidth>:ABSLimit,

136

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HTOFdm:BW<bandwidth>:BAND<band>:Y:A,

138

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HTOFdm:BW<bandwidth>:BAND<band>:Y:B,

139

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HTOFdm:BW<bandwidth>:BAND<band>:Y:C,

140

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HTOFdm:BW<bandwidth>:BAND<band>:Y:D,

141

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:HTOFdm:BW<bandwidth>:ENABle,

142

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:LOFDm:ENABle,

143

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:LOFDm:Y:A,

143

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:LOFDm:Y:B,

143

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:LOFDm:Y:C,

143

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:LOFDm:Y:D,

143

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW<bandwidth>:ABSolute:Y:A,

146

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW<bandwidth>:ABSolute:Y:B,

147

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW<bandwidth>:ABSolute:Y:C,

148

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW<bandwidth>:ABSolute:Y:D,

149

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW<bandwidth>:ABSolute:Y:E,

149

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW<bandwidth>:ABSolute:Y:F,

150

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW<bandwidth>:CA:Y:A,

151

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW<bandwidth>:CA:Y:B,

152

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW<bandwidth>:CA:Y:C,

153

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW<bandwidth>:CA:Y:D,

154

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW<bandwidth>:CA:Y:E,

155

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSMask:POFDm:BW<bandwidth>:CB:Y:A,

156

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSM

157

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSM

158

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSM

158

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSM

159

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSM

160

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSM

161

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSM

162

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSM

163

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSM

164

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSM

164

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSM

166

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSM

167

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSM

168

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSM

169

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSM

170

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:TSM

170

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:UTE

67

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIMit:UTE

67

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST,

171

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:BTYP

171

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:BWID

171

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:CMOD

171

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:CMWS

192

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:COUN

171

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:ENPo

171

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:FREQ

171

CONFigure:WLAN:MEASurement<Instance>:MEValuation:LIST:MOFF

171
CONFigure:WLAN:MEASurement<Instance>:MEValuation:REPetitio
171
CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:EV
177
CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:EV
177
CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:EV
171
CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:IQ
178
CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:MS
178
CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:PV
181
CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:SP
180
CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:TS
189
CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:UT
181
CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult[:A
182
CONFigure:WLAN:MEASurement<Instance>:MEValuation:SCONditio
183
CONFigure:WLAN:MEASurement<Instance>:MEValuation:SCOunt:MO
184
CONFigure:WLAN:MEASurement<Instance>:MEValuation:SCOunt:PV
185
CONFigure:WLAN:MEASurement<Instance>:MEValuation:SCOunt:TS
186
CONFigure:WLAN:MEASurement<Instance>:MEValuation:SFLatness
187
CONFigure:WLAN:MEASurement<Instance>:MEValuation:SMODe,
188
CONFigure:WLAN:MEASurement<Instance>:MEValuation:TOUT,
190
CONFigure:WLAN:MEASurement<Instance>:MEValuation:TSMask:AP
191
CONFigure:WLAN:MEASurement<Instance>:MEValuation:TSMask:DM
171
CONFigure:WLAN:MEASurement<Instance>:MEValuation:TSMask:MS
171
CONFigure:WLAN:MEASurement<Instance>:MEValuation:TSMask:OB
58
CONFigure:WLAN:MEASurement<instance>:MEValuation:TSMask:TR
192
CONFigure:WLAN:MEASurement<instance>:MIMO:NOANtennas,
192
CONFigure:WLAN:MEASurement<Instance>:MODE, 50
192
CONFigure:WLAN:MEASurement<Instance>:MEValuation:PVTime:REDGe,
192
CONFigure:WLAN:MEASurement<instance>:MEValuation:PVTime:RPOWer,
192
CONFigure:WLAN:MEASurement<Instance>:MEValuation:REPetitio
58
CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:EV
194
CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:EV
194
CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:EV
194
CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:IQ
194
CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:MS
194
CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:PV
194
CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:SP
194
CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:TS
194
CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult:UT
194
CONFigure:WLAN:MEASurement<Instance>:MEValuation:RESult[:A
194
CONFigure:WLAN:MEASurement<Instance>:MEValuation:SCONditio
58
CONFigure:WLAN:MEASurement<Instance>:MEValuation:SCOunt:MO
199
CONFigure:WLAN:MEASurement<Instance>:MEValuation:SCOunt:PV
199
CONFigure:WLAN:MEASurement<Instance>:MEValuation:SCOunt:TS
199
CONFigure:WLAN:MEASurement<Instance>:MEValuation:SFLatness
200
CONFigure:WLAN:MEASurement<Instance>:MEValuation:SMODe,
58
CONFigure:WLAN:MEASurement<Instance>:MEValuation:TOUT,
58
CONFigure:WLAN:MEASurement<Instance>:MEValuation:TSMask:AP
201
CONFigure:WLAN:MEASurement<Instance>:MEValuation:TSMask:DM
201
CONFigure:WLAN:MEASurement<Instance>:MEValuation:TSMask:MS
201
CONFigure:WLAN:MEASurement<Instance>:MEValuation:TSMask:OB
201
CONFigure:WLAN:MEASurement<Instance>:MEValuation:TSMask:TR
201
CONFigure:WLAN:MEASurement<instance>:MIMO:NOANtennas,
57
CONFigure:WLAN:MEASurement<Instance>:MODE, 50
CONFigure:WLAN:MEASurement<Instance>:RFSettings:ANTenna<n>
205
CONFigure:WLAN:MEASurement<Instance>:RFSettings:EATTenuati
206

CONFigure:WLAN:MEASurement<Instance>:RFSettings:ENPower, 207
CONFigure:WLAN:MEASurement<Instance>:RFSettings:FOFFset, 203
CONFigure:WLAN:MEASurement<Instance>:RFSettings:FREQuency, 208
CONFigure:WLAN:MEASurement<Instance>:RFSettings:FREQuency:MEASured, 208
CONFigure:WLAN:MEASurement<Instance>:RFSettings:FREQuency:MEASured:Chan, 210
CONFigure:WLAN:MEASurement<Instance>:RFSettings:FREQuency:MEASured:Channel, 208
CONFigure:WLAN:MEASurement<Instance>:RFSettings:LOFFset, 203
CONFigure:WLAN:MEASurement<Instance>:RFSettings:RLEVel, 211
CONFigure:WLAN:MEASurement<Instance>:RFSettings:TOFFset, 203
CONFigure:WLAN:MEASurement<Instance>:RFSettings:UMARgin, 203
CONFigure:WLAN:MEASurement<Instance>:RFSettings:UMARgin:MEASured, 212
CONFigure:WLAN:MEASurement<instance>:SMIMo:CTUPlink, 213
CONFigure:WLAN:MEASurement<instance>:TMODe:FILE:DATA, 214
CONFigure:WLAN:MEASurement<instance>:TMODe:FILE:SAVE, 214
CONFigure:WLAN:MEASurement<Instance>:TMODe:NOAPtech, 213

D

default_mode (*ScpiLogger attribute*), 867
device_name (*ScpiLogger attribute*), 867

E

error() (*ScpiLogger method*), 868

F

FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:BPOWer:AVERage, 229
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:BPOWer:CURRent, 229
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:BPOWer:MAXimum, 230
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:BPOWer:MINimum, 230
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:BPOWer:SDEViation, 231
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:CFACtor:AVERage, 231
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:CFACtor:CURRent, 232
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:CFACtor:MAXimum, 232
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:CFACtor:MINimum, 233
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:CFACtor:SDEViation, 233
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMagnitude:AVERage, 234
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMagnitude:CURRent, 235
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMagnitude:MAXimum, 235
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMagnitude:MINimum, 236
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMagnitude:SDEViation, 236
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:FERRor:AVERage, 237
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:FERRor:CURRent, 237
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:FERRor:MAXimum, 238
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:FERRor:MINimum, 238
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:FERRor:SDEViation, 239
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:AVERage, 240
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:CURRent, 240
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:MAXimum, 241
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:MINimum, 241
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:SDEViation, 242
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:AVERage, 243
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:CURRent, 243
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:MAXimum, 244
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:MINimum, 244
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:SDEViation, 245
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:SERRor:AVERage, 246
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:SERRor:CURRent, 246
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:SERRor:MAXimum, 247

FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:SDEViation, 247
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMRms:AVERage, 248
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMRms:CURRent, 249
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMRms:MAXimum, 249
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMRms:MINimum, 250
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMRms:SDEViation, 250
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMPeak:AVERage, 254
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMPeak:CURRent, 254
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMPeak:MAXimum, 255
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMPeak:MINimum, 255
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMPeak:SDEViation, 256
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMRms:AVERage, 251
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMRms:CURRent, 251
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMRms:MAXimum, 252
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMRms:MINimum, 252
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMRms:SDEViation, 253
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:GIMBalance:AVERage, 256
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:GIMBalance:CURRent, 257
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:GIMBalance:MAXimum, 257
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:GIMBalance:MINimum, 258
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:GIMBalance:SDEViation, 258
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:IQOffset:AVERage, 259
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:IQOffset:CURRent, 260
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:IQOffset:MAXimum, 260
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:IQOffset:MINimum, 261
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:IQOffset:SDEViation, 261
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:QERRor:AVERage, 262
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:SDEViation, 262
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMRms:AVERage, 263
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMRms:CURRent, 263
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMRms:MAXimum, 264
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMRms:MINimum, 265
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMRms:SDEViation, 265
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMPeak:AVERage, 266
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMPeak:CURRent, 266
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMPeak:MAXimum, 267
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMPeak:MINimum, 267
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMPeak:SDEViation, 268
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMRms:AVERage, 268
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMRms:CURRent, 269
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMRms:MAXimum, 269
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMRms:MINimum, 270
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:EVMRms:SDEViation, 271
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:GIMBalance:AVERage, 271
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:GIMBalance:CURRent, 272
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:GIMBalance:MAXimum, 272
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:GIMBalance:MINimum, 273
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:GIMBalance:SDEViation, 273
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:IQOffset:AVERage, 274
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:IQOffset:CURRent, 274
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:IQOffset:MAXimum, 275
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:IQOffset:MINimum, 276
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:IQOffset:SDEViation, 276
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:MODulation:QERRor:AVERage, 277

FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<
277
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<
278
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<
278
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<
279
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<
279
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<
280
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<
280
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<
281
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<
282
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<
282
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<
283
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<
283
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<
284
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<
284
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SEGMent<
285
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:SRELiabi
285
FETCh:WLAN:MEASurement<Instance>:MEValuation:LIST:TSMask:S
286
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:AC
286
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:AC
287
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:AC
288
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:AC
288
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:AV
289
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CP
289
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CM
290
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CM
291
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CM
292
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CM
294
295
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CM
296
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CM
297
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CM
298
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CM
299
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CM
301
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CM
303
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CM
304
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CM
305
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CM
306
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CM
307
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CM
308
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CM
309
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CM
309
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CM
310
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CM
311
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CM
312
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CM
313
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CM
315
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CM
316
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CM
318
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CM
320
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CM
323
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CM
324
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CM
325
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CM
326
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:CM
327

FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:MCHAnnel:PSTS:CURRent, 328
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:MCHAnnel:PSTS:MAXimum, 329
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:MCHAnnel:PSTS:MINimum, 330
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:MCHAnnel:PSTS:SDEViation, 331
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:MCHAnnel:SDEViation, 332
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:OFDMa:CURRent, 333
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:OFDMa:DSSS:AVERage, 336
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:OFDMa:DSSS:CURRent, 338
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:OFDMa:DSSS:MAXimum, 340
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:OFDMa:DSSS:MINimum, 342
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:OFDMa:DSSS:SDEViation, 344
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:OFDMa:EVMeasurement:AVERage, 346
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SDMA:CURRent, 347
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SDMA:EVMeasurement:MAXimum, 347
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SDMA:EVMeasurement:SDEViation, 348
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SDMA:USER<user>:AVERage, 349
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SDMA:USER<user>:CURRent, 350
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SDMA:USER<user>:MAXimum, 350
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SDMA:USER<user>:SDEViation, 351
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SDMA:USER<user>:SEValuation:AVERage, 352
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SDMA:USER<user>:SEValuation:CURRent, 353
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SDMA:USER<user>:SEValuation:MAXimum, 354
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SDMA:USER<user>:SEValuation:SDEViation, 354
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:SDMA:UINFo<user>:MAXimum, 355
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SDMA:UINFo<user>:POWer:AVERage, 358
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SDMA:UINFo<user>:POWer:CURRent, 360
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SDMA:UINFo<user>:POWer:MAXimum, 363
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SDMA:UINFo<user>:POWer:MINimum, 365
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SDMA:UINFo<user>:POWer:SDEViation, 370
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SDMA:UINFo<user>:SEValuation:AVERage, 373
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SDMA:UINFo<user>:SEValuation:CURRent, 376
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SDMA:UINFo<user>:SEValuation:MAXimum, 379
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SDMA:UINFo<user>:SEValuation:MINimum, 384
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SDMA:UINFo<user>:SEValuation:SDEViation, 387
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SDMA:UINFo<user>:SFLatness:AVERage, 389
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SDMA:UINFo<user>:SFLatness:CURRent, 390
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SDMA:UINFo<user>:SFLatness:MAXimum, 392
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SDMA:UINFo<user>:SFLatness:MINimum, 418
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SDMA:UINFo<user>:SFLatness:SDEViation, 394
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SDMA:UINFo<user>:SYMBol:CLOCk:AVERage, 397
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SDMA:UINFo<user>:SYMBol:CLOCk:CURRent, 399
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SDMA:UINFo<user>:SYMBol:CLOCk:MAXimum, 402
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SDMA:UINFo<user>:SYMBol:CLOCk:MINimum, 404
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SDMA:UINFo<user>:SYMBol:CLOCk:SDEViation, 407
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SDMA:UINFo<user>:SYMBol:TABLe:AVERage, 410
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SDMA:UINFo<user>:SYMBol:TABLe:CURRent, 412
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SDMA:UINFo<user>:SYMBol:TABLe:MAXimum, 415
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SDMA:UINFo<user>:SYMBol:TABLe:SDEViation, 421
FETCh:WLAN:MEASurement<Instance>:MEValuation:MODulation:SDMA:UINFo<user>:SYMBol:TINFo, 422
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SDMA:UINFo<user>:TMEasurement:AVERage, 423
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SDMA:UINFo<user>:TMEasurement:CURRent, 424
FETCh:WLAN:MEASurement<instance>:MEValuation:MODulation:SDMA:UINFo<user>:TMEasurement:MAXimum, 424

FETCh:WLAN:MEASurement<instance>:MEValuation:PVTime:WLAN:MEASurement<instance>:MEValuation:SFLatness:AVERage
425
FETCh:WLAN:MEASurement<instance>:MEValuation:PVTime:WLAN:MEASurement<instance>:MEValuation:SFLatness:CURRent
426
FETCh:WLAN:MEASurement<instance>:MEValuation:PVTime:WLAN:MEASurement<instance>:MEValuation:SFLatness:MAXimum
427
FETCh:WLAN:MEASurement<instance>:MEValuation:PVTime:WLAN:MEASurement<instance>:MEValuation:SFLatness:MINimum
427
FETCh:WLAN:MEASurement<instance>:MEValuation:PVTime:WLAN:MEASurement<instance>:MEValuation:SFLatness:MINimum
428
FETCh:WLAN:MEASurement<instance>:MEValuation:PVTime:WLAN:MEASurement<instance>:MEValuation:SFLatness:MINimum
429
FETCh:WLAN:MEASurement<instance>:MEValuation:PVTime:WLAN:MEASurement<instance>:MEValuation:SFLatness:MINimum
429
FETCh:WLAN:MEASurement<instance>:MEValuation:PVTime:WLAN:MEASurement<instance>:MEValuation:SFLatness:MINimum
430
FETCh:WLAN:MEASurement<instance>:MEValuation:PVTime:WLAN:MEASurement<instance>:MEValuation:SFLatness:MINimum
430
FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:WLAN:MEASurement<instance>:MEValuation:SFLatness:MINimum
431
FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:WLAN:MEASurement<instance>:MEValuation:SFLatness:MINimum
433
FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:WLAN:MEASurement<instance>:MEValuation:SFLatness:MINimum
434
FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:WLAN:MEASurement<instance>:MEValuation:SFLatness:X:A
436
FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:WLAN:MEASurement<instance>:MEValuation:SFLatness:X:C
437
FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:WLAN:MEASurement<instance>:MEValuation:SFLatness:X:M
439
FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:WLAN:MEASurement<instance>:MEValuation:SFLatness:X:M
440
FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:WLAN:MEASurement<instance>:MEValuation:SINFo:HEB:CHA
441
FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:WLAN:MEASurement<instance>:MEValuation:SINFo:HEB:CHA
443
FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:WLAN:MEASurement<instance>:MEValuation:SINFo:HEB:CHA
444
FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:WLAN:MEASurement<instance>:MEValuation:SINFo:HEB:CHA
445
FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:WLAN:MEASurement<instance>:MEValuation:SINFo:HEB:CHA
446
FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:WLAN:MEASurement<instance>:MEValuation:SINFo:HEB:CHA
447
FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:WLAN:MEASurement<instance>:MEValuation:SINFo:HEB:CHA
449
FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:WLAN:MEASurement<instance>:MEValuation:SINFo:HEB:CHA
450
FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:WLAN:MEASurement<instance>:MEValuation:SINFo:HEB:CHA
451
FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:WLAN:MEASurement<instance>:MEValuation:SINFo:HEB:CHA
453
FETCh:WLAN:MEASurement<Instance>:MEValuation:PVTime:WLAN:MEASurement<instance>:MEValuation:SINFo:HEB:CHA
454
513
514
516
518
520
522
524
526
527
528
530
531
533
534
535
536
456
457
457
458
459
460
461
461
462
463
463

FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEB:CHANnel<ch_index>:UField<usr_index>:STAid, 464
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEB:CHANnel<ch_index>:UField<usr_index>:TAIL, 465
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEB:CHANnel<ch_index>:UField<usr_index>:TXBeamform, 466
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:BDCM, 467
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:BMCS, 467
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:BSSColor, 468
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:BW, 468
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:CRC, 469
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:DOPPler, 469
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:GLTFsize, 470
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:LDPC, 470
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:NLTFsymbols, 471
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:NSBSymbols, 471
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:PEDisambig, 472
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:PEXPadding, 472
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:REServed, 473
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:SBCompress, 474
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:SPATialreuse, 474
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:STBC, 475
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:TAIL, 475
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:TXOP, 476
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HEMU:ULDL, 476
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:BEAMchange, 477
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:BSSColor, 477
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:BW, 478
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:CODing, 478
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:CRC, 479

FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:DC 480
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:DC 480
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:FO 481
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:GI 481
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:LD 482
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:MC 482
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:NS 483
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:PE 483
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:PE 484
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:RE 484
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:SP 485
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:ST 486
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:TA 486
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:TX 487
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:TX 487
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HESU:UL 488
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HETB:BS 488
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HETB:BW 489
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HETB:CR 490
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HETB:FO 490
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HETB:RE 491
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HETB:SP 492
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HETB:TA 493
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HETB:TX 493
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HTSig:A 494
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HTSig:C 495
FETCh:WLAN:MEASurement<instance>:MEValuation:SINFo:HTSig:C 495

FETCh:WLAN:MEASurement<instance>:MEValuation:SEGMent:HTSIg:MEASurement<instance>:MEValuation:SINFo:VHTSig:
496 511
FETCh:WLAN:MEASurement<instance>:MEValuation:SEGMent:HTSIg:MEASurement<instance>:MEValuation:SINFo:VHTSig:
496 511
FETCh:WLAN:MEASurement<instance>:MEValuation:SEGMent:HTSIg:MCASurement<instance>:MEValuation:SINFo:VHTSig:
497 512
FETCh:WLAN:MEASurement<instance>:MEValuation:SEGMent:HTSIg:MEASurement<Instance>:MEValuation:STATe,
497 537
FETCh:WLAN:MEASurement<instance>:MEValuation:SEGMent:HTSIg:MEASurement<Instance>:MEValuation:STATe:ALL,
498 537
FETCh:WLAN:MEASurement<instance>:MEValuation:SEGMent:HTSIg:MEASurement<Instance>:MEValuation:TRACe:CFERror
498 538
FETCh:WLAN:MEASurement<instance>:MEValuation:SEGMent:HTSIg:MEASurement<Instance>:MEValuation:TRACe:EVMagni
499 540
FETCh:WLAN:MEASurement<instance>:MEValuation:SEGMent:HTSIg:MEASurement<Instance>:MEValuation:TRACe:EVMagni
499 541
FETCh:WLAN:MEASurement<instance>:MEValuation:SEGMent:HTSIg:MEASurement<Instance>:MEValuation:TRACe:EVMagni
500 542
FETCh:WLAN:MEASurement<instance>:MEValuation:SEGMent:HTSIg:MEASurement<Instance>:MEValuation:TRACe:EVMagni
500 543
FETCh:WLAN:MEASurement<instance>:MEValuation:SEGMent:LSIGnal:MEASurement<Instance>:MEValuation:TRACe:EVMagni
501 545
FETCh:WLAN:MEASurement<instance>:MEValuation:SEGMent:LSIGnal:MEASurement<Instance>:MEValuation:TRACe:EVMagni
501 546
FETCh:WLAN:MEASurement<instance>:MEValuation:SEGMent:LSIGnal:MCASurement<instance>:MEValuation:TRACe:EVMagni
502 547
FETCh:WLAN:MEASurement<instance>:MEValuation:SEGMent:LSIGnal:MEASurement<Instance>:MEValuation:TRACe:EVMagni
503 549
FETCh:WLAN:MEASurement<instance>:MEValuation:SEGMent:LSIGnal:MCASurement<Instance>:MEValuation:TRACe:EVMagni
503 550
FETCh:WLAN:MEASurement<instance>:MEValuation:SEGMent:VHTSig:MEASurement<Instance>:MEValuation:TRACe:EVMagni
504 551
FETCh:WLAN:MEASurement<instance>:MEValuation:SEGMent:VHTSig:MEASurement<Instance>:MEValuation:TRACe:EVMagni
504 553
FETCh:WLAN:MEASurement<instance>:MEValuation:SEGMent:VHTSig:MCASurement<instance>:MEValuation:TRACe:EVMagni
505 554
FETCh:WLAN:MEASurement<instance>:MEValuation:SEGMent:VHTSig:MEASurement<Instance>:MEValuation:TRACe:EVMagni
506 556
FETCh:WLAN:MEASurement<instance>:MEValuation:SEGMent:VHTSig:MCASurement<Instance>:MEValuation:TRACe:EVMagni
506 557
FETCh:WLAN:MEASurement<instance>:MEValuation:SEGMent:VHTSig:MEASurement<Instance>:MEValuation:TRACe:EVMagni
507 558
FETCh:WLAN:MEASurement<instance>:MEValuation:SEGMent:VHTSig:MEASurement<Instance>:MEValuation:TRACe:EVMagni
507 560
FETCh:WLAN:MEASurement<instance>:MEValuation:SEGMent:VHTSig:MEASurement<instance>:MEValuation:TRACe:EVMagni
508 561
FETCh:WLAN:MEASurement<instance>:MEValuation:SEGMent:VHTSig:MEASurement<Instance>:MEValuation:TRACe:EVMagni
509 562
FETCh:WLAN:MEASurement<instance>:MEValuation:SEGMent:VHTSig:MEASurement<Instance>:MEValuation:TRACe:EVMagni
509 564
FETCh:WLAN:MEASurement<instance>:MEValuation:SEGMent:VHTSig:MCASurement<Instance>:MEValuation:TRACe:EVMagni
510 565
FETCh:WLAN:MEASurement<instance>:MEValuation:SEGMent:VHTSig:MEASurement<Instance>:MEValuation:TRACe:EVMagni
510 566

FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
567
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
569
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
570
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
571
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
572
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
573
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
574
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
576
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
577
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
578
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
579
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
580
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
581
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
582
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
583
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
584
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
586
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
587
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
588
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
589
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
591
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
592
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
593
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
594
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
595
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
596
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
597
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
598
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
599
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
601
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
602
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
603
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
605
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
606
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
608
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
609
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
611
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
612
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
613
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
615
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
616
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
617
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
619
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
620
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
621
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
622
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
624
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
625
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
626
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
627
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
629
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
630
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
631
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
633

FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
634
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
636
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatne
637
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatne
638
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatne
639
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatne
640
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatne
642
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatne
643
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatne
644
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatne
646
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatne
647
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatne
649
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatne
650
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatne
652
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatne
653
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatne
655
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatne
656
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatne
657
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatne
659
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatne
660
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatne
661
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatne
663
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatne
664
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatne
665
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatne
667
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatne
668
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatne
669
670
672
673
675
676
678
680
682
684
685
687
688
689
690
695
697
700
702
704
706
709
711
714
716
719
721
723

FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO:RXANtenna<n>:SEGMent<Seg>:AVERage, 725
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO:RXANtenna<n>:SEGMent<Seg>:CURRent, 728
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO:RXANtenna<n>:SEGMent<Seg>:MAXimum, 730
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO:RXANtenna<n>:SEGMent<Seg>:MINimum, 738
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:SEGMent<Seg>:AVERage, 739
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:SEGMent<Seg>:CURRent, 741
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:SEGMent<Seg>:MAXimum, 743
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:SEGMent<Seg>:MINimum, 733
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:ACSisoOFDM:AVERage, 734
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:ACSisoOFDM:CURRent, 735
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:ACSisoOFDM:MAXimum, 736
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:ACSisoOFDM:MINimum, 745
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:AVERage, 746
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:CURRent, 747
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:DSSS:AVERage, 748
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:DSSS:CURRent, 749
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:DSSS:MAXimum, 750
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:FREQuency, 752
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MASK, 753
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MASK:MIMO<n>, 755
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MASK:MIMO<n>:SEGMent<Seg>, 756
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MAXimum, 758
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>, 759
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>:AVERage, 760
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>:CURRent, 761
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>:MAXimum, 763
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>:MINimum, 764
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>:SEGMent<Seg>:AVERage, 766
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>:SEGMent<Seg>:CURRent, 767
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>:SEGMent<Seg>:MAXimum, 768
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>:SEGMent<Seg>:MINimum, 770
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:SEGMent<Seg>:AVERage, 771
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:SEGMent<Seg>:CURRent, 772
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:SEGMent<Seg>:MAXimum, 773
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:SEGMent<Seg>:MINimum, 775
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:UOFDM:AVERage, 777
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:UOFDM:CURRent, 778
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:UOFDM:MAXimum, 780
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:UOFDM:MINimum, 781
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:VERSion, 783
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MASK:AVERage, 784
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MASK:CURRent, 785
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MASK:FREQuency, 787
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MASK:MASK, 788
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MASK:MIMO<n>, 790
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MASK:MIMO<n>:SEGMent<Seg>, 791
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MASK:SEGMent<Seg>, 792
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MASK:SEGMent<Seg>:MIMO<n>, 794
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>:AVERage, 796
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>:CURRent, 798
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>:MAXimum, 799
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>:MINimum, 801
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>:SEGMent<Seg>:AVERage, 803
FETCh:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MIMO<n>:SEGMent<Seg>:CURRent, 804

FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO:MEASUrement<Instance>:MEValuation:TSMask:SEGMen
806
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO:MEASUrement<Instance>:MEValuation:UTERror<n>:AV
808
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO:MEASUrement<Instance>:MEValuation:UTERror<n>:CU
809
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO:MEASUrement<Instance>:MEValuation:UTERror<n>:LI
811
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO:MEASUrement<Instance>:MEValuation:UTERror<n>:MA
813
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO:MEASUrement<Instance>:MEValuation:UTERror<n>:MA
814
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO:MEASUrement<Instance>:MEValuation:UTERror<n>:MA
816
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO:MEASUrement<Instance>:MEValuation:UTERror<n>:MA
817
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO:MEASUrement<Instance>:MEValuation:UTERror<n>:MA
819
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO:MEASUrement<Instance>:MEValuation:UTERror<n>:MI
820
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:NSISo:MEASUrement<Instance>:TMODe:ANTenna<Antennas>,
822
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:NSISo:CURRent,
823
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:NSISo:MAXimum,
825
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:OBW,
826
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:OBW:MIMO<n>,
827
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:OBW:MIMO<n>:SEGMents,
828
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:OBW:SEGMents,
829
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:OFDM:AVERage,
830
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:OFDM:CURRent,
832
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:OFDM:MAXimum,
833
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:AVERage,
834
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:CURRent,
835
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:FREQuency:AVERage,
837
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:FREQuency:CURRent,
838
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:FREQuency:MAXimum,
839
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:FREQuency:MINimum,
841
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:MAXimum,
842
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO:MEASUrement<Instance>:MEValuation:TSMask:SEGMen
843
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO:MEASUrement<Instance>:MEValuation:UTERror<n>:AV
845
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO:MEASUrement<Instance>:MEValuation:UTERror<n>:CU
846
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO:MEASUrement<Instance>:MEValuation:UTERror<n>:LI
847
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO:MEASUrement<Instance>:MEValuation:UTERror<n>:MA
849
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO:MEASUrement<Instance>:MEValuation:UTERror<n>:MA
850
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO:MEASUrement<Instance>:MEValuation:UTERror<n>:MA
852
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO:MEASUrement<Instance>:MEValuation:UTERror<n>:MA
853
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO:MEASUrement<Instance>:MEValuation:UTERror<n>:MA
855
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:MIMO:MEASUrement<Instance>:MEValuation:UTERror<n>:MI
856
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:NSISo:MEASUrement<Instance>:TMODe:ANTenna<Antennas>,
858

FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:NSISo:CURRent, 868
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:NSISo:MAXimum,
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:OBW,
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:OBW:MIMO<n>,
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:OBW:MIMO<n>:SEGMents,
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:OBW:SEGMents,
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:OFDM:AVERage,
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:OFDM:CURRent,
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:OFDM:MAXimum,
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:AVERage,
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:CURRent,
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:FREQuency:AVERage,
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:FREQuency:CURRent,
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:FREQuency:MAXimum,
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:FREQuency:MINimum,
FETCh:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:MAXimum,

G

get_logging_target() (*ScpiLogger method*), 867
get_relative_timestamp() (*ScpiLogger method*), 868

I

info() (*ScpiLogger method*), 868
info_raw() (*ScpiLogger method*), 867
INITiate:WLAN:MEASurement<Instance>:MEValuation, 226
INITiate:WLAN:MEASurement<Instance>:TMODe:ANTenna<Antennas>, 858

L

log_status_check_ok (*ScpiLogger attribute*), 868
log_to_console (*ScpiLogger attribute*), 867
log_to_console_and_udp (*ScpiLogger attribute*), 867
log_to_udp (*ScpiLogger attribute*), 867

M

mode (*ScpiLogger attribute*), 867

R

READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:ACS
313
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:ACS
315
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:ACS
316

READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:MIN
318
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:MIN
320
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:MIN
323
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:OFD
324
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:OFD
325
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:OFD
326
READ:WLAN:MEASurement<instance>:MEValuation:MODulation:OFD
327
READ:WLAN:MEASurement<instance>:MEValuation:MODulation:SDE
328
READ:WLAN:MEASurement<instance>:MEValuation:MODulation:SEC
329
READ:WLAN:MEASurement<instance>:MEValuation:MODulation:SEC
330
READ:WLAN:MEASurement<instance>:MEValuation:MODulation:SEC
331
READ:WLAN:MEASurement<instance>:MEValuation:MODulation:SEC
332
READ:WLAN:MEASurement<instance>:MEValuation:MODulation:SEC
333
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:SMI
336
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:SMI
338
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:SMI
340
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:SMI
342
READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:FEDGe:A
344
READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:FEDGe:C
355
READ:WLAN:MEASurement<instance>:MEValuation:PVTime:FEDGe:M
358
READ:WLAN:MEASurement<instance>:MEValuation:PVTime:MIMO<n>
360
READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:REDGe:A
363
READ:WLAN:MEASurement<instance>:MEValuation:PVTime:REDGe:C
365
READ:WLAN:MEASurement<instance>:MEValuation:PVTime:REDGe:M
382
READ:WLAN:MEASurement<instance>:MEValuation:PVTime:TEDistr
367
READ:WLAN:MEASurement<instance>:MEValuation:PVTime:TERRor:
370
READ:WLAN:MEASurement<instance>:MEValuation:PVTime:TERRor:
373
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:MIN
376
READ:WLAN:MEASurement<instance>:MEValuation:MODulation:MIN
379
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:MIN
384
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:OFD
387
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:OFD
389
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:OFD
390
READ:WLAN:MEASurement<instance>:MEValuation:MODulation:OFD
392
READ:WLAN:MEASurement<instance>:MEValuation:MODulation:SDE
418
READ:WLAN:MEASurement<instance>:MEValuation:MODulation:SEC
394
READ:WLAN:MEASurement<instance>:MEValuation:MODulation:SEC
397
READ:WLAN:MEASurement<instance>:MEValuation:MODulation:SEC
399
READ:WLAN:MEASurement<instance>:MEValuation:MODulation:SEC
402
READ:WLAN:MEASurement<instance>:MEValuation:MODulation:SEC
404
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:SMI
407
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:SMI
410
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:SMI
412
READ:WLAN:MEASurement<Instance>:MEValuation:MODulation:SMI
415
READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:FEDGe:A
431
READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:FEDGe:C
433
READ:WLAN:MEASurement<instance>:MEValuation:PVTime:FEDGe:M
434
READ:WLAN:MEASurement<instance>:MEValuation:PVTime:MIMO<n>
436
READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:REDGe:A
437
READ:WLAN:MEASurement<instance>:MEValuation:PVTime:REDGe:C
439
READ:WLAN:MEASurement<instance>:MEValuation:PVTime:REDGe:M
440
READ:WLAN:MEASurement<instance>:MEValuation:PVTime:TEDistr
441
READ:WLAN:MEASurement<instance>:MEValuation:PVTime:TERRor:
443
READ:WLAN:MEASurement<instance>:MEValuation:PVTime:TERRor:
444

READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:WEAR:MEASurement<Instance>:MEValuation:TRACe:EVMagnit
445 542
READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:WEAR:MEASurement<Instance>:MEValuation:TRACe:EVMagnit
446 543
READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:WEAR:MEASurement<Instance>:MEValuation:TRACe:EVMagnit
447 545
READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:WEAR:MEASurement<Instance>:MEValuation:TRACe:EVMagnit
449 546
READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:WEAR:MEASurement<Instance>:MEValuation:TRACe:EVMagnit
450 547
READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:WEAR:MEASurement<Instance>:MEValuation:TRACe:EVMagnit
451 549
READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:WEAR:MEASurement<Instance>:MEValuation:TRACe:EVMagnit
453 550
READ:WLAN:MEASurement<Instance>:MEValuation:PVTime:WEAR:MEASurement<Instance>:MEValuation:TRACe:EVMagnit
454 551
READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnit
513 553
READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnit
514 554
READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnit
516 556
READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnit
518 557
READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnit
520 558
READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnit
522 560
READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnit
524 561
READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnit
526 562
READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnit
527 564
READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnit
528 565
READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnit
530 566
READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnit
531 567
READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnit
533 569
READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnit
534 570
READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnit
535 571
READ:WLAN:MEASurement<instance>:MEValuation:SFLatness:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnit
536 572
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:CFERror:MEASurement<Instance>:MEValuation:TRACe:EVMagnit
538 573
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:ACSimbol:AVERage,READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnit
540 574
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:ACSimbol:CURRent,READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMagnit
541 576

READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMAll:MEASurement:CARRier:CURRent,
577
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMAll:MEASurement:CARRier:MAXimum,
578
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMAll:MEASurement:SYMBol:AVERage,
579
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMAll:MEASurement:SYMBol:CURRent,
580
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMAll:MEASurement:SYMBol:MAXimum,
581
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMAll:MEASurement:SYMBol:MIMO<In>:AVERage,
582
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMAll:MEASurement:SYMBol:MIMO<In>:CURRent,
583
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMAll:MEASurement:SYMBol:MIMO<In>:MAXimum,
584
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMAll:MEASurement:SYMBol:MIMO<In>:SEGMent:AVERage,
586
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMAll:MEASurement:SYMBol:MIMO<In>:SEGMent:CURRent,
587
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMAll:MEASurement:SYMBol:MIMO<In>:SEGMent:MAXimum,
588
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMAll:MEASurement:SYMBol:MIMO<In>:SEGMent:MINimum,
589
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:EVMAll:MEASurement:SYMBol:MINimum,
591
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:IQCons:MEASurement<Instance>:MEValuation:TRACe:PVTime:M
592
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:IQCons:MEASurement<Instance>:MEValuation:TRACe:PVTime:M
593
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTAll:MEASurement<Instance>:MEValuation:TRACe:PVTime:M
594
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTAll:MEASurement<Instance>:MEValuation:TRACe:PVTime:M
595
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTAll:MEASurement:AVERage,
596
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTAll:MEASurement:CURRent,
597
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTAll:MEASurement:MAXimum,
598
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTAll:MEASurement:MIMO<In>:AVERage,
599
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTAll:MEASurement:MIMO<In>:CURRent,
601
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTAll:MEASurement:MIMO<In>:MAXimum,
602
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTAll:MEASurement:MIMO<In>:MINimum,
603
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTAll:MEASurement:MIMO<In>:SEGMent:AVERage,
605
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTAll:MEASurement:MIMO<In>:SEGMent:CURRent,
606
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTAll:MEASurement:MIMO<In>:SEGMent:MAXimum,
608
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
609
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
611
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
612
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
613
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
615
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
616
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
617
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
619
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
620
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
621
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
622
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
624
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
625
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
626
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
627
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
629
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
630
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
631
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
633
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
634
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
636
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
637
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
638
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
639
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
640
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
642
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:PVTime:
643

READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MAXimum, 644
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:MINimum, 646
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SEGMent<Instance>:AVERage, 647
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SEGMent<Instance>:CURRent, 649
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SEGMent<Instance>:MAXimum, 650
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SEGMent<Instance>:MINimum, 652
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SEGMent<Instance>:TIME, 653
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:TIME, 655
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness[:AVERage], 656
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness[:CURRent], 657
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness[:MAXimum], 659
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness[:MINimum], 660
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness[:TIME], 661
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:TIME, 663
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TIME, 664
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TIME:SEGMent<Instance>:AVERage, 665
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TIME:SEGMent<Instance>:CURRent, 667
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TIME:SEGMent<Instance>:MAXimum, 668
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TIME:SEGMent<Instance>:MINimum, 669
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TIME:SEGMent<Instance>:TIME, 670
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TIME:TIME, 672
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SCARrier:AVERage, 673
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SCARrier:CURRent, 675
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SCARrier:MAXimum, 676
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SCARrier:MINimum, 678
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SCARrier:SEGMent<Instance>:AVERage, 680
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SCARrier:SEGMent<Instance>:CURRent, 682
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SCARrier:SEGMent<Instance>:MAXimum, 684
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SCARrier:SEGMent<Instance>:MINimum, 685
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SCARrier:SEGMent<Instance>:MEValuation, 687
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SCARrier:SEGMent<Instance>:TIME, 688
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SCARrier:TIME, 689
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SEGMent<Instance>:AVERage, 690
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SEGMent<Instance>:CURRent, 695
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SEGMent<Instance>:MAXimum, 697
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SEGMent<Instance>:MINimum, 700
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SEGMent<Instance>:TIME, 702
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:TIME, 704
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TIME, 706
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TIME:SEGMent<Instance>:AVERage, 709
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TIME:SEGMent<Instance>:CURRent, 711
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TIME:SEGMent<Instance>:MAXimum, 714
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TIME:SEGMent<Instance>:MINimum, 716
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TIME:SEGMent<Instance>:TIME, 719
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TIME:TIME, 721
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SCARrier:SEGMent<Instance>:AVERage, 723
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SCARrier:SEGMent<Instance>:CURRent, 725
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SCARrier:SEGMent<Instance>:MAXimum, 728
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SCARrier:SEGMent<Instance>:MINimum, 730
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness:SCARrier:SEGMent<Instance>:TIME, 733

READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness[:OFDM]:CURRent, 734
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness[:OFDM]:MAXimum, 735
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness[:OFDM]:MINimum, 736
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TDRamp, 745
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:ACSiso: 746
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:DSSS:AV 747
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:DSSS:CU 748
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:DSSS:MA 749
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:FREQuer 750
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:FREQuer 752
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:FREQuer 753
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:MAXimum 755
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:MIMO<n> 756
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:MIMO<n> 758
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:MIMO<n> 759
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:MIMO<n> 760
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:MIMO<n> 761
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:MIMO<n> 763
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:MIMO<n> 764
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:MIMO<n> 766
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:MIMO<n> 767
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:MIMO<n> 768
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:MIMO<n> 770
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:MIMO<n> 771
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:MIMO<n> 772
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:MIMO<n> 773
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:WCAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n> 775
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness[:OFDM]:CURRent, 777
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness[:OFDM]:MAXimum, 778
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:SFLatness[:OFDM]:MINimum, 780
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TDRamp, 781
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:DSSS:AV 783
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:DSSS:CU 784
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:DSSS:MA 785
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:FREQuer 787
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:FREQuer 788
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:FREQuer 790
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:FREQuer 791
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:MAXimum 792
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:MIMO<n> 794
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:MIMO<n> 796
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:MIMO<n> 798
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:MIMO<n> 799
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:MIMO<n> 801
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:MIMO<n> 803
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:MIMO<n> 804
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:MIMO<n> 806
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:MIMO<n> 808
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:MIMO<n> 809
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:MIMO<n> 811
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:MIMO<n> 813
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:MIMO<n> 814
READ:WLAN:MEASurement<Instance>:MEValuation:TRACe:TSMask:MEASurement<Instance>:MEValuation:TSMask:MIMO<n> 816
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:WCAN:MEASurement<Instance>:MEValuation:TSMask:MIMO<n> 817

READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:MINimum, 819
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:MARGin, 820
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:MSI:MAXimum, 822
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:NSI:CURRent, 823
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:OBW:MAXimum, 825
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:OBW:SEGMents, 826
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:OBW:SEGMents:AVERage, 827
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:OBW:SEGMents:CURRent, 828
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:OBW:SEGMents:MAXimum, 829
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:OFDM:AVERage, 830
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:OFDM:CURRent, 832
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:OFDM:MAXimum, 833
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:AVERage, 834
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:CURRent, 835
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:FREQuency:AVERage, 837
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:FREQuency:CURRent, 838
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:FREQuency:MAXimum, 839
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:FREQuency:MINimum, 841
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:MAXimum, 842
READ:WLAN:MEASurement<Instance>:MEValuation:TSMask:SEGMents:MINimum, 843
READ:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:AVERage, 845
READ:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:CURRent, 846
READ:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:LIMit, 847
READ:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MARGin:AVERage, 849
READ:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MARGin:CURRent, 850
READ:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MARGin:MAXimum, 852
READ:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MARGin:MINimum, 853
READ:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MAXimum, 855
READ:WLAN:MEASurement<Instance>:MEValuation:UTERror<n>:MINimum, 856
READ:WLAN:MEASurement<Instance>:TMODe:ANTenna<Antennas>, 858
Reset_Statistical_Count() (*ScpiLogger method*), 868
ROUTe:WLAN:MEASurement<Instance>, 215
ROUTe:WLAN:MEASurement<Instance>:CATalog:SCENario, 216
ROUTe:WLAN:MEASurement<Instance>:SCENario, 217
ROUTe:WLAN:MEASurement<Instance>:SCENario:CSPath, 217
ROUTe:WLAN:MEASurement<Instance>:SCENario:SALone, 218
ROUTe:WLAN:MEASurement<Instance>:SCENario:SMI<nr>, 219
ROUTe:WLAN:MEASurement<Instance>:SCENario:SMIMo<PathCount>, 220
ROUTe:WLAN:MEASurement<Instance>:SCENario:TMIMo<PathCount>, 221
ROUTe:WLAN:MEASurement<Instance>:SMIMo, 215

S

ScpiLogger (*class in RsCMPX_WlanMeas.Internal.ScpiLogger*), 867
set_format_string() (*ScpiLogger method*), 868
set_logging_target() (*ScpiLogger method*), 867
set_logging_target_global() (*ScpiLogger method*), 867
set_relative_timestamp() (*ScpiLogger method*), 868
set_relative_timestamp_now() (*ScpiLogger method*), 868
STOP:WLAN:MEASurement<Instance>:MEValuation, 226

T

target_auto_flushing (*ScpiLogger attribute*), 868
TRIGger:WLAN:MEASurement<Instance>:MEValuation:CATalog:SOURce, 225
TRIGger:WLAN:MEASurement<Instance>:MEValuation:MGAP, 222
TRIGger:WLAN:MEASurement<Instance>:MEValuation:OFFSet, 222
TRIGger:WLAN:MEASurement<Instance>:MEValuation:SLOPe, 222
TRIGger:WLAN:MEASurement<Instance>:MEValuation:SOURce, 222
TRIGger:WLAN:MEASurement<Instance>:MEValuation:THReshold, 222
TRIGger:WLAN:MEASurement<Instance>:MEValuation:TOUT, 222

U

udp_port (*ScpiLogger attribute*), 868