RsCmwLteMeas

Release 4.0.110.26

Rohde & Schwarz

Apr 17, 2024

CONTENTS:


CMW

CHAPTER

ONE

REVISION HISTORY

1.1 RsCmwLteMeas

Rohde & Schwarz CMW LTE Measurement RsCmwLteMeas instrument driver.

Basic Hello-World code:

```
from RsCmwLteMeas import *

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

Supported instruments: CMW500, CMW100

The package is hosted here: https://pypi.org/project/RsCmwLteMeas/

Documentation: https://RsCmwLteMeas.readthedocs.io/

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

1.1.1 Version history

Release Notes:

Latest release notes summary: Update for FW 4.0.110

Version 4.0.110

- Update for FW 4.0.110

Version 3.8.xx2

- Fixed several misspelled arguments and command headers

Version 3.8.xx1

- Bluetooth and WLAN update for FW versions 3.8.xxx

Version 3.7.xx8

- Added documentation on ReadTheDocs

Version 3.7.xx7

- Added 3G measurement subsystems RsCmwGsmMeas, RsCmwCdma2kMeas, RsCmwEvdoMeas, RsCmwWcdmaMeas
- Added new data types for commands accepting numbers or ON/OFF:
- int or bool
- float or bool

Version 3.7.xx6

- Added new UDF integer number recognition

Version 3.7.xx5

- Added RsCmwDau

Version 3.7.xx4

- Fixed several interface names
- New release for CMW Base 3.7.90
- New release for CMW Bluetooth 3.7.90

Version 3.7.xx3

- Second release of the CMW python drivers packet
- New core component RsInstrument
- Previously, the groups starting with CATalog: e.g. 'CATalog:SIGNaling:TOPology:PLMN' were reordered to 'SIGNaling:TOPology:PLMN:CATALOG' give more contextual meaning to the method/property name. This is now reverted back, since it was hard to find the desired functionality.
- Reorganized Utilities interface to sub-groups

Version 3.7.xx2

- Fixed some misspeling errors
- Changed enum and repCap types names
- All the assemblies are signed with Rohde & Schwarz signature

Version 1.0.0.0

- First released version

CHAPTER

TWO

GETTING STARTED

2.1 Introduction

RsCmwLteMeas 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

RsCmwLteMeas 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 RsCmwLteMeas`

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 `RsCmwLteMeas` 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 RsCmwLteMeas offline:

- Download this python script (**Save target as**): rsinstrument_offline_install.py This installs all the preconditions that the RsCmwLteMeas needs.
- Execute the script in your offline computer (supported is python 3.6 or newer)
- Download the RsCmwLteMeas package to your computer from the pypi.org: https://pypi.org/project/RsCmwLteMeas/#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\RsCmwLteMeas-4.0.110.26.tar`

2.3 Finding Available Instruments

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

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

from RsCmwLteMeas import *

# Use the instr_list string items as resource names in the RsCmwLteMeas constructor
instr_list = RsCmwLteMeas.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 RsCmwLteMeas 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 = RsCmwLteMeas.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

RsCmwLteMeas 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 RsCmwLteMeas object. Below, is a simple Hello World example. Different resource names are examples for different physical interfaces.

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

- Installed RsCmwLteMeas Python module Version 4.0.110 or newer from pypi.org
- Installed VISA, for example R&S Visa 5.12 or newer
"""

from RsCmwLteMeas import *

# A good practice is to assure that you have a certain minimum version installed
RsCmwLteMeas.assert_minimum_version('4.0.110')
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 = RsCmwLteMeas(resource_string_1)

idn = driver.utilities.query_str('*IDN?')
print(f"\nHello, I am: '{idn}'")
print(f'RsCmwLteMeas 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; RsCmwLteMeas 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 = RsCmwLteMeas('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 RsCmwLteMeas 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 RsCmwLteMeas allows you to choose which VISA to use:

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

from RsCmwLteMeas import *

# Force use of the Rs Visa. For NI Visa, use the "SelectVisa='ni'"
driver = RsCmwLteMeas('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, RsCmwLteMeas has something for you too - **no Visa installation raw LAN socket**:

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

from RsCmwLteMeas import *

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

More `option_string` tokens are separated by comma:

```
driver = RsCmwLteMeas('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 RsCmwLteMeas objects:

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

from RsCmwLteMeas import *

driver1 = RsCmwLteMeas('TCPIP::192.168.56.101::INSTR', True, True)
driver2 = RsCmwLteMeas.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 RsCmwLteMeas 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 RsCmwLteMeas import *

driver = RsCmwLteMeas('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 RsCmwLteMeas import *

driver = RsCmwLteMeas('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 RsCmwLteMeas raises an exception. Speaking of exceptions, an important feature of the RsCmwLteMeas 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 RsCmwLteMeas import *

driver = RsCmwLteMeas('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

RsCmwLteMeas 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 RsCmwLteMeas 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 RsCmwLteMeas import *

driver = None
# Try-catch for initialization. If an error occures, the ResourceError is raised
try:
```

(continues on next page)

(continued from previous page)

```
    driver = RsCmwLteMeas('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 RsCmwLteMeas exceptions
    print(e.args[0])
    print('Some other RsCmwLteMeas 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 RsCmwLteMeas, 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 RsCmwLteMeas 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 RsCmwLteMeas has this covered. And, this feature is quite universal - not just for big files transfer, but for any data in both directions.

RsCmwLteMeas 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 RsCmwLteMeas 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 = RsCmwLteMeas('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
# 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 RsCmwLteMeas 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, RsCmwLteMeas 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 RsCmwLteMeas object
"""

import threading
from RsCmwLteMeas import *


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


driver = RsCmwLteMeas('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')
```

(continues on next page)

(continued from previous page)

```
# 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 RsCmwLteMeas objects with shared session
"""

import threading
from RsCmwLteMeas import *


def execute(session: RsCmwLteMeas, 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 = RsCmwLteMeas('TCPIP::192.168.56.101::INSTR')
driver2 = RsCmwLteMeas.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()

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()
```

(continues on next page)

(continued from previous page)

```
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 RsCmwLteMeas takes care of it for you. The text below describes this scenario.

Run the following example:

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

import threading
from RsCmwLteMeas import *


def execute(session: RsCmwLteMeas, 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 = RsCmwLteMeas('TCPIP::192.168.56.101::INSTR')
driver2 = RsCmwLteMeas('TCPIP::192.168.56.101::INSTR')
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,))
```

(continues on next page)

(continued from previous page)

```
    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 RsCmwLteMeas import *

driver = RsCmwLteMeas('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
```

(continues on next page)

(continued from previous page)

```
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 = RsCmwLteMeas('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 RsCmwLteMeas import *

driver = RsCmwLteMeas('TCPIP::192.168.1.101::INSTR')

# 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
```

(continues on next page)

(continued from previous page)

```
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
"""

from RsCmwLteMeas import *

driver = RsCmwLteMeas('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')
```

(continues on next page)

(continued from previous page)

```
# 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 Band

```
# First value:
value = enums.Band.OB1
# Last value:
value = enums.Band.OB9
# All values (63x):
OB1 | OB10 | OB11 | OB12 | OB13 | OB14 | OB15 | OB16
OB17 | OB18 | OB19 | OB2 | OB20 | OB21 | OB22 | OB23
OB24 | OB25 | OB250 | OB26 | OB27 | OB28 | OB3 | OB30
OB31 | OB33 | OB34 | OB35 | OB36 | OB37 | OB38 | OB39
OB4 | OB40 | OB41 | OB42 | OB43 | OB44 | OB45 | OB46
OB47 | OB48 | OB49 | OB5 | OB50 | OB51 | OB52 | OB53
OB6 | OB65 | OB66 | OB68 | OB7 | OB70 | OB71 | OB72
OB73 | OB74 | OB8 | OB85 | OB87 | OB88 | OB9
```

3.2 CarrAggrLocalOscLocation

```
# Example value:
value = enums.CarrAggrLocalOscLocation.AUTO
# All values (3x):
AUTO | CACB | CECC
```

3.3 CarrAggrMaping

```
# First value:
value = enums.CarrAggrMaping.INV
# Last value:
value = enums.CarrAggrMaping.SCC7
# All values (9x):
INV | PCC | SCC1 | SCC2 | SCC3 | SCC4 | SCC5 | SCC6
SCC7
```

3.4 CarrAggrMode

```
# Example value:
value = enums.CarrAggrMode.ICD
# All values (4x):
ICD | ICE | INTRaband | OFF
```

3.5 ChannelBandwidth

```
# Example value:
value = enums.ChannelBandwidth.B014
# All values (6x):
B014 | B030 | B050 | B100 | B150 | B200
```

3.6 ChannelTypeDetection

```
# Example value:
value = enums.ChannelTypeDetection.AUTO
# All values (3x):
AUTO | PUCCh | PUSCh
```

3.7 ChannelTypeVewFilter

```
# Example value:
value = enums.ChannelTypeVewFilter.OFF
# All values (4x):
OFF | ON | PUCCh | PUSCh
```

3.8 CmwsConnector

```
# First value:
value = enums.CmwsConnector.R11
# Last value:
value = enums.CmwsConnector.RB8
# All values (48x):
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
```

3.9 CyclicPrefix

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

3.10 DuplexMode

```
# Example value:
value = enums.DuplexMode.FDD
# All values (2x):
FDD | TDD
```

3.11 FrameStructure

```
# Example value:
value = enums.FrameStructure.T1
# All values (2x):
T1 | T2
```

3.12 LaggingExclPeriod

```
# Example value:
value = enums.LaggingExclPeriod.MS05
# All values (3x):
MS05 | MS25 | OFF
```

3.13 LeadingExclPeriod

```
# Example value:
value = enums.LeadingExclPeriod.MS25
# All values (2x):
MS25 | OFF
```

3.14 ListMode

```
# Example value:
value = enums.ListMode.ONCE
# All values (2x):
ONCE | SEGMent
```

3.15 LocalOscLocation

```
# Example value:
value = enums.LocalOscLocation.CCB
# All values (2x):
CCB | CN
```

3.16 LowHigh

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

3.17 MeasCarrier

```
# Example value:
value = enums.MeasCarrier.PCC
# All values (2x):
PCC | SCC1
```

3.18 MeasCarrierEnhanced

```
# Example value:
value = enums.MeasCarrierEnhanced.CC1
# All values (4x):
CC1 | CC2 | CC3 | CC4
```

3.19 MeasFilter

```
# Example value:
value = enums.MeasFilter.BANDpass
# All values (2x):
BANDpass | GAUSs
```

3.20 MeasurementMode

```
# Example value:
value = enums.MeasurementMode.MELMode
# All values (3x):
MELMode | NORMal | TMODe
```

3.21 MeasureSlot

```
# Example value:
value = enums.MeasureSlot.ALL
# All values (3x):
ALL | MS0 | MS1
```

3.22 MevAcquisitionMode

```
# Example value:
value = enums.MevAcquisitionMode.SLOT
# All values (2x):
SLOT | SUBFrame
```

3.23 ModScheme

```
# Example value:
value = enums.ModScheme.AUTO
# All values (5x):
AUTO | Q16 | Q256 | Q64 | QPSK
```

3.24 Modulation

```
# Example value:
value = enums.Modulation.Q16
# All values (4x):
Q16 | Q256 | Q64 | QPSK
```

3.25 Nbandwidth

```
# Example value:
value = enums.Nbandwidth.M010
# All values (4x):
M010 | M020 | M040 | M080
```

3.26 NetworkSigValue

```
# First value:
value = enums.NetworkSigValue.NS01
# Last value:
value = enums.NetworkSigValue.NS32
# All values (32x):
NS01 | NS02 | NS03 | NS04 | NS05 | NS06 | NS07 | NS08
NS09 | NS10 | NS11 | NS12 | NS13 | NS14 | NS15 | NS16
NS17 | NS18 | NS19 | NS20 | NS21 | NS22 | NS23 | NS24
NS25 | NS26 | NS27 | NS28 | NS29 | NS30 | NS31 | NS32
```

3.27 NetworkSigValueNoCarrAggr

```
# First value:
value = enums.NetworkSigValueNoCarrAggr.NS01
# Last value:
value = enums.NetworkSigValueNoCarrAggr.NS99
# All values (288x):
NS01 | NS02 | NS03 | NS04 | NS05 | NS06 | NS07 | NS08
NS09 | NS10 | NS100 | NS101 | NS102 | NS103 | NS104 | NS105
NS106 | NS107 | NS108 | NS109 | NS11 | NS110 | NS111 | NS112
NS113 | NS114 | NS115 | NS116 | NS117 | NS118 | NS119 | NS12
NS120 | NS121 | NS122 | NS123 | NS124 | NS125 | NS126 | NS127
NS128 | NS129 | NS13 | NS130 | NS131 | NS132 | NS133 | NS134
NS135 | NS136 | NS137 | NS138 | NS139 | NS14 | NS140 | NS141
NS142 | NS143 | NS144 | NS145 | NS146 | NS147 | NS148 | NS149
NS15 | NS150 | NS151 | NS152 | NS153 | NS154 | NS155 | NS156
NS157 | NS158 | NS159 | NS16 | NS160 | NS161 | NS162 | NS163
NS164 | NS165 | NS166 | NS167 | NS168 | NS169 | NS17 | NS170
NS171 | NS172 | NS173 | NS174 | NS175 | NS176 | NS177 | NS178
```

(continues on next page)

(continued from previous page)

```
NS179 | NS18 | NS180 | NS181 | NS182 | NS183 | NS184 | NS185
NS186 | NS187 | NS188 | NS189 | NS19 | NS190 | NS191 | NS192
NS193 | NS194 | NS195 | NS196 | NS197 | NS198 | NS199 | NS20
NS200 | NS201 | NS202 | NS203 | NS204 | NS205 | NS206 | NS207
NS208 | NS209 | NS21 | NS210 | NS211 | NS212 | NS213 | NS214
NS215 | NS216 | NS217 | NS218 | NS219 | NS22 | NS220 | NS221
NS222 | NS223 | NS224 | NS225 | NS226 | NS227 | NS228 | NS229
NS23 | NS230 | NS231 | NS232 | NS233 | NS234 | NS235 | NS236
NS237 | NS238 | NS239 | NS24 | NS240 | NS241 | NS242 | NS243
NS244 | NS245 | NS246 | NS247 | NS248 | NS249 | NS25 | NS250
NS251 | NS252 | NS253 | NS254 | NS255 | NS256 | NS257 | NS258
NS259 | NS26 | NS260 | NS261 | NS262 | NS263 | NS264 | NS265
NS266 | NS267 | NS268 | NS269 | NS27 | NS270 | NS271 | NS272
NS273 | NS274 | NS275 | NS276 | NS277 | NS278 | NS279 | NS28
NS280 | NS281 | NS282 | NS283 | NS284 | NS285 | NS286 | NS287
NS288 | NS29 | NS30 | NS31 | NS32 | NS33 | NS34 | NS35
NS36 | NS37 | NS38 | NS39 | NS40 | NS41 | NS42 | NS43
NS44 | NS45 | NS46 | NS47 | NS48 | NS49 | NS50 | NS51
NS52 | NS53 | NS54 | NS55 | NS56 | NS57 | NS58 | NS59
NS60 | NS61 | NS62 | NS63 | NS64 | NS65 | NS66 | NS67
NS68 | NS69 | NS70 | NS71 | NS72 | NS73 | NS74 | NS75
NS76 | NS77 | NS78 | NS79 | NS80 | NS81 | NS82 | NS83
NS84 | NS85 | NS86 | NS87 | NS88 | NS89 | NS90 | NS91
NS92 | NS93 | NS94 | NS95 | NS96 | NS97 | NS98 | NS99
```

3.28 ParameterSetMode

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

3.29 Path

```
# Example value:
value = enums.Path.NETWork
# All values (2x):
NETWork | STANdalone
```

3.30 PeriodPreamble

```
# Example value:
value = enums.PeriodPreamble.MS05
# All values (3x):
MS05 | MS10 | MS20
```

3.31 PucchFormat

```
# Example value:
value = enums.PucchFormat.F1
# All values (7x):
F1 | F1A | F1B | F2 | F2A | F2B | F3
```

3.32 RbTableChannelType

```
# Example value:
value = enums.RbTableChannelType.DL
# All values (8x):
DL | NONE | PSBCh | PSCCh | PSSCh | PUCCh | PUSCh | SSUB
```

3.33 Rbw

```
# Example value:
value = enums.Rbw.K030
# All values (3x):
K030 | K100 | M1
```

3.34 RbwExtended

```
# Example value:
value = enums.RbwExtended.K030
# All values (6x):
K030 | K050 | K100 | K150 | K200 | M1
```

3.35 Repeat

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

3.36 ResourceState

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

3.37 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
```

3.38 RetriggerFlag

```
# Example value:
value = enums.RetriggerFlag.IFPNarrow
# All values (4x):
IFPNarrow | IFPower | OFF | ON
```

3.39 RxConnector

```
# First value:
value = enums.RxConnector.I11I
# Last value:
value = enums.RxConnector.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
```

(continues on next page)

(continued from previous page)

```
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
```

3.40 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
```

3.41 Scenario

```
# Example value:
value = enums.Scenario.CSPath
# All values (4x):
CSPath | MAPRotocol | NAV | SALone
```

3.42 SegmentChannelTypeExtended

```
# Example value:
value = enums.SegmentChannelTypeExtended.AUTO
# All values (6x):
AUTO | PSBCh | PSCCh | PSSCh | PUCCh | PUSCh
```

3.43 Sharing

```
# Example value:
value = enums.Sharing.FSHared
# All values (3x):
FSHared | NSHared | OCONnection
```

3.44 SidelinkChannelType

```
# Example value:
value = enums.SidelinkChannelType.PSBCh
# All values (3x):
PSBCh | PSCCh | PSSCh
```

3.45 SignalSlope

```
# Example value:
value = enums.SignalSlope.FEDGe
# All values (2x):
FEDGe | REDGe
```

3.46 SignalType

```
# Example value:
value = enums.SignalType.SL
# All values (2x):
SL | UL
```

3.47 StopCondition

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

3.48 SyncMode

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

3.49 TargetMainState

```
# Example value:
value = enums.TargetMainState.OFF
# All values (3x):
OFF | RDY | RUN
```

3.50 TargetSyncState

```
# Example value:
value = enums.TargetSyncState.ADJusted
# All values (2x):
ADJusted | PENDing
```

3.51 TimeMask

```
# Example value:
value = enums.TimeMask.GOO
# All values (3x):
GOO | PPSRs | SBLanking
```

3.52 TraceSelect

```
# Example value:
value = enums.TraceSelect.AVERage
# All values (3x):
AVERage | CURRent | MAXimum
```

3.53 UplinkChannelType

```
# Example value:
value = enums.UplinkChannelType.PUCCh
# All values (2x):
PUCCh | PUSCh
```

CHAPTER

FOUR

REPCAPS

4.1 Instance (Global)

```
# Setting:
driver.repcap_instance_set(repcap.Instance.Inst1)
# Range:
Inst1 .. Inst16
# All values (16x):
Inst1 | Inst2 | Inst3 | Inst4 | Inst5 | Inst6 | Inst7 | Inst8
Inst9 | Inst10 | Inst11 | Inst12 | Inst13 | Inst14 | Inst15 | Inst16
```

4.2 AbsMarker

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

4.3 Area

```
# First value:
value = repcap.Area.Nr1
# Range:
Nr1 .. Nr12
# All values (12x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
Nr9 | Nr10 | Nr11 | Nr12
```

4.4 CarrierComponent

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

4.5 ChannelBw

```
# First value:
value = repcap.ChannelBw.Bw14
# Range:
Bw14 .. Bw200
# All values (6x):
Bw14 | Bw30 | Bw50 | Bw100 | Bw150 | Bw200
```

4.6 DeltaMarker

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

4.7 Difference

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

4.8 EutraBand

```
# First value:
value = repcap.EutraBand.Nr30
# Values (2x):
Nr30 | Nr50
```

4.9 FirstChannelBw

```
# First value:
value = repcap.FirstChannelBw.Bw100
# Values (3x):
Bw100 | Bw150 | Bw200
```

4.10 Limit

```
# First value:
value = repcap.Limit.Nr1
# Range:
Nr1 .. Nr12
# All values (12x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
Nr9 | Nr10 | Nr11 | Nr12
```

4.11 MaxRange

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

4.12 MinRange

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

4.13 Preamble

```
# First value:
value = repcap.Preamble.Nr1
# Range:
Nr1 .. Nr400
# All values (400x):
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
```

(continues on next page)

(continued from previous page)

```
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
Nr145 | Nr146 | Nr147 | Nr148 | Nr149 | Nr150 | Nr151 | Nr152
Nr153 | Nr154 | Nr155 | Nr156 | Nr157 | Nr158 | Nr159 | Nr160
Nr161 | Nr162 | Nr163 | Nr164 | Nr165 | Nr166 | Nr167 | Nr168
Nr169 | Nr170 | Nr171 | Nr172 | Nr173 | Nr174 | Nr175 | Nr176
Nr177 | Nr178 | Nr179 | Nr180 | Nr181 | Nr182 | Nr183 | Nr184
Nr185 | Nr186 | Nr187 | Nr188 | Nr189 | Nr190 | Nr191 | Nr192
Nr193 | Nr194 | Nr195 | Nr196 | Nr197 | Nr198 | Nr199 | Nr200
Nr201 | Nr202 | Nr203 | Nr204 | Nr205 | Nr206 | Nr207 | Nr208
Nr209 | Nr210 | Nr211 | Nr212 | Nr213 | Nr214 | Nr215 | Nr216
Nr217 | Nr218 | Nr219 | Nr220 | Nr221 | Nr222 | Nr223 | Nr224
Nr225 | Nr226 | Nr227 | Nr228 | Nr229 | Nr230 | Nr231 | Nr232
Nr233 | Nr234 | Nr235 | Nr236 | Nr237 | Nr238 | Nr239 | Nr240
Nr241 | Nr242 | Nr243 | Nr244 | Nr245 | Nr246 | Nr247 | Nr248
Nr249 | Nr250 | Nr251 | Nr252 | Nr253 | Nr254 | Nr255 | Nr256
Nr257 | Nr258 | Nr259 | Nr260 | Nr261 | Nr262 | Nr263 | Nr264
Nr265 | Nr266 | Nr267 | Nr268 | Nr269 | Nr270 | Nr271 | Nr272
Nr273 | Nr274 | Nr275 | Nr276 | Nr277 | Nr278 | Nr279 | Nr280
Nr281 | Nr282 | Nr283 | Nr284 | Nr285 | Nr286 | Nr287 | Nr288
Nr289 | Nr290 | Nr291 | Nr292 | Nr293 | Nr294 | Nr295 | Nr296
Nr297 | Nr298 | Nr299 | Nr300 | Nr301 | Nr302 | Nr303 | Nr304
Nr305 | Nr306 | Nr307 | Nr308 | Nr309 | Nr310 | Nr311 | Nr312
Nr313 | Nr314 | Nr315 | Nr316 | Nr317 | Nr318 | Nr319 | Nr320
Nr321 | Nr322 | Nr323 | Nr324 | Nr325 | Nr326 | Nr327 | Nr328
Nr329 | Nr330 | Nr331 | Nr332 | Nr333 | Nr334 | Nr335 | Nr336
Nr337 | Nr338 | Nr339 | Nr340 | Nr341 | Nr342 | Nr343 | Nr344
Nr345 | Nr346 | Nr347 | Nr348 | Nr349 | Nr350 | Nr351 | Nr352
Nr353 | Nr354 | Nr355 | Nr356 | Nr357 | Nr358 | Nr359 | Nr360
Nr361 | Nr362 | Nr363 | Nr364 | Nr365 | Nr366 | Nr367 | Nr368
Nr369 | Nr370 | Nr371 | Nr372 | Nr373 | Nr374 | Nr375 | Nr376
Nr377 | Nr378 | Nr379 | Nr380 | Nr381 | Nr382 | Nr383 | Nr384
Nr385 | Nr386 | Nr387 | Nr388 | Nr389 | Nr390 | Nr391 | Nr392
Nr393 | Nr394 | Nr395 | Nr396 | Nr397 | Nr398 | Nr399 | Nr400
```

4.14 PreambleFormat

```
# First value:
value = repcap.PreambleFormat.Fmt1
# Range:
Fmt1 .. Fmt5
# All values (5x):
Fmt1 | Fmt2 | Fmt3 | Fmt4 | Fmt5
```

4.15 QAMmodOrder

```
# First value:
value = repcap.QAMmodOrder.Qam16
# Values (3x):
Qam16 | Qam64 | Qam256
```

4.16 RBcount

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

4.17 RBoffset

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

4.18 RBWkHz

```
# First value:
value = repcap.RBWkHz.Rbw30
# Range:
Rbw30 .. Rbw1000
# All values (6x):
Rbw30 | Rbw50 | Rbw100 | Rbw150 | Rbw200 | Rbw1000
```

4.19 Ripple

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

4.20 SecondaryCC

```
# First value:
value = repcap.SecondaryCC.CC1
# Range:
CC1 .. CC7
# All values (7x):
CC1 | CC2 | CC3 | CC4 | CC5 | CC6 | CC7
```

4.21 SecondChannelBw

```
# First value:
value = repcap.SecondChannelBw.Bw50
# Values (4x):
Bw50 | Bw100 | Bw150 | Bw200
```

4.22 Segment

```
# First value:
value = repcap.Segment.Nr1
# Range:
Nr1 .. Nr128
# All values (128x):
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
```

4.23 Table

```
# First value:
value = repcap.Table.Nr1
# Range:
Nr1 .. Nr5
# All values (5x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5
```

4.24 ThirdChannelBw

```
# First value:
value = repcap.ThirdChannelBw.Bw100
# Values (3x):
Bw100 | Bw150 | Bw200
```

4.25 UtraAdjChannel

```
# First value:
value = repcap.UtraAdjChannel.Ch1
# Values (2x):
Ch1 | Ch2
```

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

RSCMWLTEMEAS API STRUCTURE

Global RepCaps

```
driver = RsCmwLteMeas('TCPIP::192.168.2.101::hislip0')
# Instance range: Inst1 .. Inst16
rc = driver.repcap_instance_get()
driver.repcap_instance_set(repcap.Instance.Inst1)
```

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

890 total commands, 7 Subgroups, 0 group commands

Initializes new RsCmwLteMeas 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 RsCmwLteMeas.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 RsCmwLteMeas session.

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

Creates a new RsCmwLteMeas 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

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:BAND
CONFigure:LTE:MEASurement<Instance>:SPATh
CONFigure:LTE:MEASurement<Instance>:STYPe
CONFigure:LTE:MEASurement<Instance>:DMODe
CONFigure:LTE:MEASurement<Instance>:FSTRucture
```

class ConfigureCls

Configure commands group definition. 209 total commands, 9 Subgroups, 5 group commands

get_band() → Band

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:BAND
value: enums.Band = driver.configure.get_band()
```

Selects the operating band (OB) .

INTRO_CMD_HELP: The allowed input range has dependencies:

- FDD UL: OB1 | … | OB28 | OB30 | OB31 | OB65 | OB66 | OB68 | OB70 | … | OB74 | OB85 | OB87 | OB88
- TDD UL: OB33 | … | OB45 | OB48 | OB50 | … | OB53 | OB250
- Sidelink: OB47

INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:LTE:SIGN<i>[:PCC]:BAND
- CONFigure:LTE:SIGN<i>:SCC<c>:BAND

return

band: OB1 to OB250, see list above

get_dmode() → DuplexMode

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:DMODe
value: enums.DuplexMode = driver.configure.get_dmode()
```

Selects the duplex mode of the LTE signal: FDD or TDD.

INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:LTE:SIGN<i>[:PCC]:DMODe
- CONFigure:LTE:SIGN<i>:SCC<c>:DMODe
- CONFigure:LTE:SIGN<i>[:PCC]:DMODe:UCSPecific

return

mode: FDD | TDD

get_fstructure() → FrameStructure

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:FSTRucture
value: enums.FrameStructure = driver.configure.get_fstructure()
```

Queries the frame structure type of the LTE signal. The value depends on the duplex mode (method RsCmwLteMeas.Configure. dmode) .

return
frame_structure: T1 | T2 T1: Type 1, FDD signal T2: Type 2, TDD signal

get_spath() → Path

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:SPATh
value: enums.Path = driver.configure.get_spath()
```

No command help available

return
path: No help available

get_stype() → SignalType

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:STYPe
value: enums.SignalType = driver.configure.get_stype()
```

Selects the type of the measured signal.

return
signal_type: UL | SL UL: LTE uplink signal with PUSCH or PUCCH SL: V2X sidelink signal with PSSCH and PSCCH

set_band(*band: Band*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:BAND
driver.configure.set_band(band = enums.Band.OB1)
```

Selects the operating band (OB) .
INTRO_CMD_HELP: The allowed input range has dependencies:

- FDD UL: OB1 | … | OB28 | OB30 | OB31 | OB65 | OB66 | OB68 | OB70 | … | OB74 | OB85 | OB87 | OB88
- TDD UL: OB33 | … | OB45 | OB48 | OB50 | … | OB53 | OB250
- Sidelink: OB47

INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:LTE:SIGN<i>[:PCC]:BAND
- CONFigure:LTE:SIGN<i>:SCC<c>:BAND

param band
OB1 to OB250, see list above

set_dmode(*mode: DuplexMode*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:DMODe
driver.configure.set_dmode(mode = enums.DuplexMode.FDD)
```

Selects the duplex mode of the LTE signal: FDD or TDD.
INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:LTE:SIGN<i>[:PCC]:DMODe
- CONFigure:LTE:SIGN<i>:SCC<c>:DMODe
- CONFigure:LTE:SIGN<i>[:PCC]:DMODe:UCSPecific

param mode
FDD | TDD

`set_spath`(*path: Path*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:SPATh
driver.configure.set_spath(path = enums.Path.NETWork)
```

No command help available

param path
No help available

`set_stype`(*signal_type: SignalType*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:STYPe
driver.configure.set_stype(signal_type = enums.SignalType.SL)
```

Selects the type of the measured signal.

param signal_type
UL | SL UL: LTE uplink signal with PUSCH or PUCCH SL: V2X sidelink signal with PSSCH and PSCCH

Cloning the Group

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

Subgroups

6.1.1 CarrierAggregation

`class CarrierAggregationCls`

CarrierAggregation commands group definition. 11 total commands, 6 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.1.1.1 ChannelBw

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:CAGGregation:CBANdwidth:AGGRegated
```

class ChannelBwCls

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

get_aggregated() → float

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:CAGGregation:CBANdwidth:AGGRegated
value: float = driver.configure.carrierAggregation.channelBw.get_aggregated()
```

Queries the width of the aggregated channel bandwidth.

return
ch_bandwidth: float Unit: Hz

6.1.1.2 Frequency

class FrequencyCls

Frequency 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.carrierAggregation.frequency.clone()
```

Subgroups

6.1.1.2.1 Aggregated

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:CAGGregation:FREQuency:AGGRegated:LOW
CONFigure:LTE:MEASurement<Instance>:CAGGregation:FREQuency:AGGRegated:CENTer
CONFigure:LTE:MEASurement<Instance>:CAGGregation:FREQuency:AGGRegated:HIGH
```

class AggregatedCls

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

get_center() → float

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:CAGGregation:FREQuency:AGGRegated:CENTer
value: float = driver.configure.carrierAggregation.frequency.aggregated.get_
↪center()
```

Queries the center frequency of the aggregated bandwidth.

return
frequency_center: float Unit: Hz

get_high() → float

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:CAGGregation:FREQuency:AGGRegated:HIGH
value: float = driver.configure.carrierAggregation.frequency.aggregated.get_
↪high()
```

Queries the upper edge of the aggregated bandwidth.

return
frequency_high: float Unit: Hz

get_low() → float

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:CAGGregation:FREQuency:AGGRegated:LOW
value: float = driver.configure.carrierAggregation.frequency.aggregated.get_
↪low()
```

Queries the lower edge of the aggregated bandwidth.

return
frequency_low: float Unit: Hz

6.1.1.3 Maping

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:CAGGregation:MAPing:PCC
CONFigure:LTE:MEASurement<Instance>:CAGGregation:MAPing
```

class MapingCls

Maping commands group definition. 3 total commands, 1 Subgroups, 2 group commands

class ValueStruct

Structure for reading output parameters. Fields:

- Cc_1: enums.CarrAggrMaping: INV | PCC | SCC1 | SCC2 | SCC3 | SCC4 | SCC5 | SCC6 | SCC7 Carrier mapped to CC1
- Cc_2: enums.CarrAggrMaping: INV | PCC | SCC1 | SCC2 | SCC3 | SCC4 | SCC5 | SCC6 | SCC7 Carrier mapped to CC2
- Cc_3: enums.CarrAggrMaping: INV | PCC | SCC1 | SCC2 | SCC3 | SCC4 | SCC5 | SCC6 | SCC7 Carrier mapped to CC3

- Cc_4: enums.CarrAggrMaping: INV | PCC | SCC1 | SCC2 | SCC3 | SCC4 | SCC5 | SCC6 | SCC7 Carrier mapped to CC4

get_pcc() → str

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:CAGGregation:MAPing:PCC
value: str = driver.configure.carrierAggregation.maping.get_pcc()
```

This command is only relevant for combined signal path measurements with contiguous uplink CA. It queries to which CC the PCC is mapped. The measurement identifies the aggregated carriers as CC1 to CC4. The signaling application uses PCC and SCC<n>.

return
cc: string Examples: 'CC1', 'CC2', 'INV' 'INV' means that the PCC is not contained in the measured set of aggregated uplink carriers.

get_value() → ValueStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:CAGGregation:MAPing
value: ValueStruct = driver.configure.carrierAggregation.maping.get_value()
```

This command is only relevant for combined signal path measurements with contiguous uplink CA. It queries which carriers are mapped to CC1 to CC4. The measurement identifies the aggregated carriers as CC1 to CC4. The signaling application uses PCC and SCC<n>. A returned INV means that no carrier is mapped to the CC.

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.configure.carrierAggregation.maping.clone()
```

Subgroups

6.1.1.3.1 Scc<SecondaryCC>

RepCap Settings

```
# Range: CC1 .. CC7
rc = driver.configure.carrierAggregation.maping.scc.repcap_secondaryCC_get()
driver.configure.carrierAggregation.maping.scc.repcap_secondaryCC_set(repcap.SecondaryCC.
↪CC1)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:CAGGregation:MAPing:SCC<Carrier>
```

class SccCls

Scc commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: SecondaryCC, default value after init: SecondaryCC.CC1

get(*secondaryCC=SecondaryCC.Default*) → str

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:CAGGregation:MAPing:SCC<Carrier>
value: str = driver.configure.carrierAggregation.maping.scc.get(secondaryCC =␣
↪repcap.SecondaryCC.Default)
```

This command is only relevant for combined signal path measurements with contiguous uplink CA. It queries to which CC the SCC<n> is mapped. The measurement identifies the aggregated carriers as CC1 to CC4. The signaling application uses PCC and SCC<n>.

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

return
cc: string Examples: 'CC1', 'CC2', 'INV' 'INV' means that the SCCn is not contained in the measured set of aggregated uplink carriers.

Cloning the Group

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

6.1.1.4 Mcarrier

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:CAGGregation:MCARrier:ENHanced
```

class McarrierCls

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

get_enhanced() → MeasCarrierEnhanced

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:CAGGregation:MCARrier:ENHanced
value: enums.MeasCarrierEnhanced = driver.configure.carrierAggregation.mcarrier.
↪get_enhanced()
```

Selects a component carrier for single-carrier measurements.

return
meas_carrier: CC1 | CC2 | CC3 | CC4

set_enhanced(*meas_carrier: MeasCarrierEnhanced*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:CAGGregation:MCARrier:ENHanced
driver.configure.carrierAggregation.mcarrier.set_enhanced(meas_carrier = enums.
↪MeasCarrierEnhanced.CC1)
```

Selects a component carrier for single-carrier measurements.

param meas_carrier
CC1 | CC2 | CC3 | CC4

6.1.1.5 Mode

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:CAGGregation:MODE:CSPath
CONFigure:LTE:MEASurement<Instance>:CAGGregation:MODE
```

class ModeCls

Mode commands group definition. 2 total commands, 0 Subgroups, 2 group commands

get_combined_signal_path() → CarrAggrMode

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:CAGGregation:MODE:CSPath
value: enums.CarrAggrMode = driver.configure.carrierAggregation.mode.get_
↪combined_signal_path()
```

Queries the carrier aggregation mode in the CSP scenario. The mode is configured indirectly via method RsCmwLteMeas.Route. Scenario.CombinedSignalPath.set.

return
ca_mode: OFF | INTRaband | ICD | ICE OFF: no carrier aggregation INTRaband: intra-band contiguous CA (BW class B & C) ICD: intra-band contiguous CA (BW class D) ICE: intra-band contiguous CA (BW class E)

get_value() → CarrAggrMode

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:CAGGregation:MODE
value: enums.CarrAggrMode = driver.configure.carrierAggregation.mode.get_value()
```

Selects how many component carriers with intra-band contiguous aggregation are measured. For the combined signal path scenario, use method RsCmwLteMeas.Route.Scenario.CombinedSignalPath.set.

return
ca_mode: OFF | INTRaband | ICD | ICE OFF: only one carrier is measured INTRaband: two carriers (BW class B & C) ICD: three carriers (BW class D) ICE: four carriers (BW class E)

set_value(*ca_mode: CarrAggrMode*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:CAGGregation:MODE
driver.configure.carrierAggregation.mode.set_value(ca_mode = enums.CarrAggrMode.
↪ICD)
```

Selects how many component carriers with intra-band contiguous aggregation are measured. For the combined signal path scenario, use method RsCmwLteMeas.Route.Scenario.CombinedSignalPath.set.

param ca_mode
OFF | INTRaband | ICD | ICE OFF: only one carrier is measured INTRaband: two carriers (BW class B & C) ICD: three carriers (BW class D) ICE: four carriers (BW class E)

6.1.1.6 Scc<SecondaryCC>

RepCap Settings

```
# Range: CC1 .. CC7
rc = driver.configure.carrierAggregation.scc.repcap_secondaryCC_get()
driver.configure.carrierAggregation.scc.repcap_secondaryCC_set(repcap.SecondaryCC.CC1)
```

class SccCls

Scc commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability: SecondaryCC, default value after init: SecondaryCC.CC1

Cloning the Group

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

Subgroups

6.1.1.6.1 AcSpacing

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:CAGGregation[:SCC<Nr>]:ACSPacing
```

class AcSpacingCls

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

set(*secondaryCC=SecondaryCC.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:CAGGregation[:SCC<Nr>]:ACSPacing
driver.configure.carrierAggregation.scc.acSpacing.set(secondaryCC = repcap.
↪SecondaryCC.Default)
```

Adjusts the component carrier frequencies, so that the carriers are aggregated contiguously. For the combined signal path scenario, use CONFigure:LTE:SIGN<i>:CAGGregation:SET.

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

set_with_opc(*secondaryCC=SecondaryCC.Default*, *opc_timeout_ms: int = -1*) → None

6.1.2 Cc<CarrierComponent>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.configure.cc.repcap_carrierComponent_get()
driver.configure.cc.repcap_carrierComponent_set(repcap.CarrierComponent.Nr1)
```

class CcCls

Cc commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability: CarrierComponent, default value after init: CarrierComponent.Nr1

Cloning the Group

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

Subgroups

6.1.2.1 ChannelBw

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:CC<Nr>:CBANdwidth
```

class ChannelBwCls

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

get(*carrierComponent=CarrierComponent.Default*) → ChannelBandwidth

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:CC<Nr>:CBANdwidth
value: enums.ChannelBandwidth = driver.configure.cc.channelBw.
↪get(carrierComponent = repcap.CarrierComponent.Default)
```

Selects the channel bandwidth of component carrier CC<no>. Without carrier aggregation, you can omit <no>.

INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:LTE:SIGN<i>:CELL:BANDwidth[:PCC]:DL
- CONFigure:LTE:SIGN<i>:CELL:BANDwidth:SCC<c>:DL

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
channel_bw: B014 | B030 | B050 | B100 | B150 | B200 B014: 1.4 MHz B030: 3 MHz B050: 5 MHz B100: 10 MHz B150: 15 MHz B200: 20 MHz

set(*channel_bw: ChannelBandwidth*, *carrierComponent=CarrierComponent.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:CC<Nr>:CBANdwidth
driver.configure.cc.channelBw.set(channel_bw = enums.ChannelBandwidth.B014,␣
↪carrierComponent = repcap.CarrierComponent.Default)
```

Selects the channel bandwidth of component carrier CC<no>. Without carrier aggregation, you can omit <no>.
INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:LTE:SIGN<i>:CELL:BANDwidth[:PCC]:DL
- CONFigure:LTE:SIGN<i>:CELL:BANDwidth:SCC<c>:DL

param channel_bw
B014 | B030 | B050 | B100 | B150 | B200 B014: 1.4 MHz B030: 3 MHz B050: 5 MHz B100: 10 MHz B150: 15 MHz B200: 20 MHz

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

6.1.3 Emtc

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:EMTC:ENABle
CONFigure:LTE:MEASurement<instance>:EMTC:MB<number>
CONFigure:LTE:MEASurement<Instance>:EMTC:NBANd
```

class EmtcCls

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

get_enable() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:EMTC:ENABle
value: bool = driver.configure.emtc.get_enable()
```

Enables or disables eMTC. For the combined signal path scenario, use CONFigure:LTE:SIGN<i>[:PCC]:EMTC:ENABle.

return
enable: OFF | ON

get_mb() → bool

```
# SCPI: CONFigure:LTE:MEASurement<instance>:EMTC:MB<number>
value: bool = driver.configure.emtc.get_mb()
```

Selects the maximum eMTC bandwidth. For the combined signal path scenario, use CONFigure:LTE:SIGN<i>[:PCC]:EMTC:MB<number>.

return
enable: OFF | ON OFF: Max bandwidth 1.4 MHz ON: Max bandwidth 5 MHz

get_nband() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:EMTC:NBANd
value: int = driver.configure.emtc.get_nband()
```

Selects the narrowband used for eMTC.

return
number: numeric The maximum depends on the channel BW, see 'RB allocation, narrowbands and widebands for eMTC'. Range: 0 to 15

set_enable(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:EMTC:ENABle
driver.configure.emtc.set_enable(enable = False)
```

Enables or disables eMTC. For the combined signal path scenario, use CONFigure:LTE:SIGN<i>[:PCC]:EMTC:ENABle.

param enable
OFF | ON

set_mb(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<instance>:EMTC:MB<number>
driver.configure.emtc.set_mb(enable = False)
```

Selects the maximum eMTC bandwidth. For the combined signal path scenario, use CONFigure:LTE:SIGN<i>[:PCC]:EMTC:MB<number>.

param enable
OFF | ON OFF: Max bandwidth 1.4 MHz ON: Max bandwidth 5 MHz

set_nband(*number: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:EMTC:NBANd
driver.configure.emtc.set_nband(number = 1)
```

Selects the narrowband used for eMTC.

param number
numeric The maximum depends on the channel BW, see 'RB allocation, narrowbands and widebands for eMTC'. Range: 0 to 15

6.1.4 MultiEval

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:TOUT
CONFigure:LTE:MEASurement<Instance>:MEValuation:MMODe
CONFigure:LTE:MEASurement<Instance>:MEValuation:REPetition
CONFigure:LTE:MEASurement<Instance>:MEValuation:SCONdition
CONFigure:LTE:MEASurement<Instance>:MEValuation:ULDL
CONFigure:LTE:MEASurement<Instance>:MEValuation:SSUBframe
CONFigure:LTE:MEASurement<Instance>:MEValuation:MOEXception
CONFigure:LTE:MEASurement<Instance>:MEValuation:CPRefix
CONFigure:LTE:MEASurement<Instance>:MEValuation:CTYPe
```

(continues on next page)

(continued from previous page)

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:SCTYpe
CONFigure:LTE:MEASurement<Instance>:MEValuation:PSEarch
CONFigure:LTE:MEASurement<Instance>:MEValuation:PFORmat
CONFigure:LTE:MEASurement<Instance>:MEValuation:NVFilter
CONFigure:LTE:MEASurement<Instance>:MEValuation:ORVFilter
CONFigure:LTE:MEASurement<Instance>:MEValuation:CTVFilter
CONFigure:LTE:MEASurement<Instance>:MEValuation:DSSPusch
CONFigure:LTE:MEASurement<Instance>:MEValuation:GHOPping
CONFigure:LTE:MEASurement<Instance>:MEValuation:MSLot
```

class MultiEvalCls

MultiEval commands group definition. 138 total commands, 16 Subgroups, 18 group commands

get_cprefix() → CyclicPrefix

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:CPRefix
value: enums.CyclicPrefix = driver.configure.multiEval.get_cprefix()
```

Selects the type of cyclic prefix of the LTE signal. For the combined signal path scenario, use CONFigure:LTE:SIGN<i>:CELL:CPRefix.

return
cyclic_prefix: NORMal | EXTended

get_ctv_filter() → ChannelTypeVewFilter

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:CTVFilter
value: enums.ChannelTypeVewFilter = driver.configure.multiEval.get_ctv_filter()
```

Specifies, enables or disables the channel type view filter. If the filter is active, only slots with detected channel type PUSCH or PUCCH are measured.

return
channel_type: PUSCh | PUCCh | ON | OFF PUSCh: measure only PUSCH PUCCh: measure only PUCCH ON: enable the filter OFF: disable the filter

get_ctype() → ChannelTypeDetection

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:CTYPe
value: enums.ChannelTypeDetection = driver.configure.multiEval.get_ctype()
```

Configures the channel type detection for uplink measurements.

return
channel_type: AUTO | PUSCh | PUCCh Automatic detection of channel type or manual selection

get_dss_pusch() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:DSSPusch
value: int = driver.configure.multiEval.get_dss_pusch()
```

Specifies the delta sequence shift value (ss) used to calculate the sequence shift pattern for PUSCH.

return
delta_seq_sh_pusch: integer Range: 0 to 29

get_ghopping() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:GHOPping
value: bool = driver.configure.multiEval.get_ghopping()
```

Specifies whether group hopping is used or not. For the combined signal path scenario, use CONFigure:LTE:SIGN<i>:CONNection:GHOPping.

return
value: OFF | ON

get_mmode() → MeasurementMode

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:MMODe
value: enums.MeasurementMode = driver.configure.multiEval.get_mmode()
```

Selects the measurement mode.

return
measurement_mode: NORMal | TMODe | MELMode NORMal: normal mode TMODe: TPC mode MELMode: multi-evaluation list mode

get_mo_exception() → bool

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

Specifies whether measurement results identified as faulty or inaccurate are rejected.

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

get_mslot() → MeasureSlot

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:MSLot
value: enums.MeasureSlot = driver.configure.multiEval.get_mslot()
```

Selects which slots of the 'Measure Subframe' are measured.

return
measure_slot: MS0 | MS1 | ALL MS0: slot number 0 only MS1: slot number 1 only ALL: both slots

get_nvfilter() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:NVFilter
value: int or bool = driver.configure.multiEval.get_nvfilter()
```

Specifies, enables or disables the number of resource blocks (NRB) view filter. If the filter is active, only slots with a matching number of allocated resource blocks are measured. Within the indicated input range, only specific numbers are allowed as defined in 3GPP TS 36.211. For details, see 'Resources in time and frequency domain'.

return
nrb_view_filter: (integer or boolean) numeric | ON | OFF Number of allocated resource blocks Range: 1 to 100 ON | OFF enables or disables the filter.

get_orv_filter() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:ORVFilter
value: int or bool = driver.configure.multiEval.get_orv_filter()
```

Specifies, enables or disables the RB offset view filter. If the filter is active, only slots with a matching number of RB offset are measured. The indicated input range applies to a 20-MHz channel bandwidth. The maximum value depends on the bandwidth (maximum number of RBs minus one) .

return
offset_rb: (integer or boolean) numeric | ON | OFF Offset of the first allocated RB Range: 0 to 99 ON | OFF enables or disables the filter.

get_peak_search() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:PSEarch
value: bool = driver.configure.multiEval.get_peak_search()
```

No command help available

return
pucch_search: No help available

get_pformat() → PucchFormat

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:PFORmat
value: enums.PucchFormat = driver.configure.multiEval.get_pformat()
```

Specifies the PUCCH format (only relevant for signals containing a PUCCH) . The formats are defined in 3GPP TS 36.211.

return
pucch_format: F1 | F1A | F1B | F2 | F2A | F2B | F3

get_repetition() → Repeat

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:REPetition
value: enums.Repeat = driver.configure.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 | CONTinuous SINGleshot: Single-shot measurement CONTinuous: Continuous measurement

get_sch_type() → SidelinkChannelType

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:SCTYpe
value: enums.SidelinkChannelType = driver.configure.multiEval.get_sch_type()
```

Configures the channel type for modulation results of sidelink measurements.

return
channel_type: PSSCh | PSCCh

get_scondition() → StopCondition

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:SCONdition
value: enums.StopCondition = driver.configure.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 | SLFail NONE: Continue measurement irrespective of the limit check. SLFail: Stop measurement on limit failure.

get_ssubframe() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:SSUBframe
value: int = driver.configure.multiEval.get_ssubframe()
```

Selects a special subframe configuration, defining the inner structure of special subframes. This parameter is only relevant for frame structure 'Type 2' (method RsCmwLteMeas.Configure.fstructure) . The special subframe configurations are defined in 3GPP TS 36.211, chapter 4, 'Frame Structure'.

INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:LTE:SIGN<i>:CELL[:PCC]:SSUBframe
- CONFigure:LTE:SIGN<i>:CELL:SCC<c>:SSUBframe
- CONFigure:LTE:SIGN<i>:CELL:TDD:SPECific

return
special_subframe: integer Range: 0 to 8

get_timeout() → float

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:TOUT
value: float = driver.configure.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.

return
timeout: numeric Unit: s

get_ul_dl() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:ULDL
value: int = driver.configure.multiEval.get_ul_dl()
```

Selects an UL-DL configuration, defining the combination of uplink, downlink and special subframes within a radio frame. This parameter is only relevant for frame structure 'Type 2' (method RsCmwLteMeas.Configure.fstructure) . The UL-DL configurations are defined in 3GPP TS 36.211, chapter 4, 'Frame Structure'.

INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:LTE:SIGN<i>:CELL[:PCC]:ULDL
- CONFigure:LTE:SIGN<i>:CELL:SCC<c>:ULDL
- CONFigure:LTE:SIGN<i>:CELL:TDD:SPECific

return
uplink_downlink: integer Range: 0 to 6

set_cprefix(*cyclic_prefix: CyclicPrefix*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:CPRefix
driver.configure.multiEval.set_cprefix(cyclic_prefix = enums.CyclicPrefix.
↪EXTended)
```

Selects the type of cyclic prefix of the LTE signal. For the combined signal path scenario, use CONFigure:LTE:SIGN<i>:CELL:CPRefix.

param cyclic_prefix
NORMal | EXTended

set_ctv_filter(*channel_type: ChannelTypeVewFilter*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:CTVFilter
driver.configure.multiEval.set_ctv_filter(channel_type = enums.
↪ChannelTypeVewFilter.OFF)
```

Specifies, enables or disables the channel type view filter. If the filter is active, only slots with detected channel type PUSCH or PUCCH are measured.

param channel_type
PUSCh | PUCCh | ON | OFF PUSCh: measure only PUSCH PUCCh: measure only PUCCH ON: enable the filter OFF: disable the filter

set_ctype(*channel_type: ChannelTypeDetection*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:CTYPe
driver.configure.multiEval.set_ctype(channel_type = enums.ChannelTypeDetection.
↪AUTO)
```

Configures the channel type detection for uplink measurements.

param channel_type
AUTO | PUSCh | PUCCh Automatic detection of channel type or manual selection

set_dss_pusch(*delta_seq_sh_pusch: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:DSSPusch
driver.configure.multiEval.set_dss_pusch(delta_seq_sh_pusch = 1)
```

Specifies the delta sequence shift value (ss) used to calculate the sequence shift pattern for PUSCH.

param delta_seq_sh_pusch
integer Range: 0 to 29

set_ghopping(*value: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:GHOPping
driver.configure.multiEval.set_ghopping(value = False)
```

Specifies whether group hopping is used or not. For the combined signal path scenario, use CONFigure:LTE:SIGN<i>:CONNection:GHOPping.

param value
OFF | ON

set_mmode(*measurement_mode: MeasurementMode*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:MMODe
driver.configure.multiEval.set_mmode(measurement_mode = enums.MeasurementMode.
↪MELMode)
```

Selects the measurement mode.

param measurement_mode
NORMal | TMODe | MELMode NORMal: normal mode TMODe: TPC mode MELMode: multi-evaluation list mode

set_mo_exception(*meas_on_exception: bool*) → None

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

Specifies whether measurement results identified as faulty or inaccurate are rejected.

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

set_mslot(*measure_slot: MeasureSlot*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:MSLot
driver.configure.multiEval.set_mslot(measure_slot = enums.MeasureSlot.ALL)
```

Selects which slots of the 'Measure Subframe' are measured.

param measure_slot
MS0 | MS1 | ALL MS0: slot number 0 only MS1: slot number 1 only ALL: both slots

set_nvfilter(*nrb_view_filter: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:NVFilter
driver.configure.multiEval.set_nvfilter(nrb_view_filter = 1)
```

Specifies, enables or disables the number of resource blocks (NRB) view filter. If the filter is active, only slots with a matching number of allocated resource blocks are measured. Within the indicated input range, only specific numbers are allowed as defined in 3GPP TS 36.211. For details, see 'Resources in time and frequency domain'.

param nrb_view_filter
(integer or boolean) numeric | ON | OFF Number of allocated resource blocks Range: 1 to 100 ON | OFF enables or disables the filter.

set_orv_filter(*offset_rb: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:ORVFilter
driver.configure.multiEval.set_orv_filter(offset_rb = 1)
```

Specifies, enables or disables the RB offset view filter. If the filter is active, only slots with a matching number of RB offset are measured. The indicated input range applies to a 20-MHz channel bandwidth. The maximum value depends on the bandwidth (maximum number of RBs minus one) .

param offset_rb
(integer or boolean) numeric | ON | OFF Offset of the first allocated RB Range: 0 to 99 ON | OFF enables or disables the filter.

set_peak_search(*pucch_search: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:PSEarch
driver.configure.multiEval.set_peak_search(pucch_search = False)
```

No command help available

param pucch_search
No help available

set_pformat(*pucch_format: PucchFormat*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:PFORmat
driver.configure.multiEval.set_pformat(pucch_format = enums.PucchFormat.F1)
```

Specifies the PUCCH format (only relevant for signals containing a PUCCH) . The formats are defined in 3GPP TS 36.211.

param pucch_format
F1 | F1A | F1B | F2 | F2A | F2B | F3

set_repetition(*repetition: Repeat*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:REPetition
driver.configure.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 | CONTinuous SINGleshot: Single-shot measurement CONTinuous: Continuous measurement

set_sch_type(*channel_type: SidelinkChannelType*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:SCTYpe
driver.configure.multiEval.set_sch_type(channel_type = enums.
↪SidelinkChannelType.PSBCh)
```

Configures the channel type for modulation results of sidelink measurements.

param channel_type
PSSCh | PSCCh

set_scondition(*stop_condition: StopCondition*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:SCONdition
driver.configure.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 | SLFail NONE: Continue measurement irrespective of the limit check. SLFail: Stop measurement on limit failure.

set_ssubframe(*special_subframe: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:SSUBframe
driver.configure.multiEval.set_ssubframe(special_subframe = 1)
```

Selects a special subframe configuration, defining the inner structure of special subframes. This parameter is only relevant for frame structure 'Type 2' (method RsCmwLteMeas.Configure.fstructure) . The special subframe configurations are defined in 3GPP TS 36.211, chapter 4, 'Frame Structure'.

INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:LTE:SIGN<i>:CELL[:PCC]:SSUBframe
- CONFigure:LTE:SIGN<i>:CELL:SCC<c>:SSUBframe
- CONFigure:LTE:SIGN<i>:CELL:TDD:SPECific

param special_subframe
integer Range: 0 to 8

set_timeout(*timeout: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:TOUT
driver.configure.multiEval.set_timeout(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.

param timeout
numeric Unit: s

set_ul_dl(*uplink_downlink: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:ULDL
driver.configure.multiEval.set_ul_dl(uplink_downlink = 1)
```

Selects an UL-DL configuration, defining the combination of uplink, downlink and special subframes within a radio frame. This parameter is only relevant for frame structure 'Type 2' (method RsCmwLteMeas.Configure.fstructure) . The UL-DL configurations are defined in 3GPP TS 36.211, chapter 4, 'Frame Structure'.

INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:LTE:SIGN<i>:CELL[:PCC]:ULDL
- CONFigure:LTE:SIGN<i>:CELL:SCC<c>:ULDL
- CONFigure:LTE:SIGN<i>:CELL:TDD:SPECific

param uplink_downlink
integer Range: 0 to 6

Cloning the Group

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

Subgroups

6.1.4.1 Bler

class BlerCls

Bler 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.multiEval.bler.clone()
```

Subgroups

6.1.4.1.1 Sframes

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:BLER:SFRames
```

class SframesCls

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

class SframesStruct

Response structure. Fields:

- Sub_Frames: int: integer Number of subframes to be measured Range: 1 subframe to 200E+3 subframes
- Sched_Subfr_Per_Fr: int: integer Number of scheduled subframes per radio frame in the generated downlink signal Range: 1 subframe to 10 subframes

get() → SframesStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:BLER:SFRames
value: SframesStruct = driver.configure.multiEval.bler.sframes.get()
```

Specifies the statistic count (number of measured subframes) and the number of scheduled subframes per radio frame for the BLER measurement. BLER is a single shot measurement.

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

set(*sub_frames: int*, *sched_subfr_per_fr: int = None*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:BLER:SFRames
driver.configure.multiEval.bler.sframes.set(sub_frames = 1, sched_subfr_per_fr␣
↪= 1)
```

Specifies the statistic count (number of measured subframes) and the number of scheduled subframes per radio frame for the BLER measurement. BLER is a single shot measurement.

param sub_frames
integer Number of subframes to be measured Range: 1 subframe to 200E+3 subframes

param sched_subfr_per_fr
integer Number of scheduled subframes per radio frame in the generated downlink signal Range: 1 subframe to 10 subframes

6.1.4.2 Cc<CarrierComponent>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.configure.multiEval.cc.repcap_carrierComponent_get()
driver.configure.multiEval.cc.repcap_carrierComponent_set(repcap.CarrierComponent.Nr1)
```

class CcCls

Cc commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability: CarrierComponent, default value after init: CarrierComponent.Nr1

Cloning the Group

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

Subgroups

6.1.4.2.1 PlcId

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:CC<Nr>:PLCid
```

class PlcIdCls

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

get(*carrierComponent=CarrierComponent.Default*) → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:CC<Nr>:PLCid
value: int = driver.configure.multiEval.cc.plcId.get(carrierComponent = repcap.
↪CarrierComponent.Default)
```

Specifies the physical layer cell ID of component carrier CC<no>. Without carrier aggregation, you can omit <no>.

INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:LTE:SIGN<i>:CELL[:PCC]:PCID
- CONFigure:LTE:SIGN<i>:CELL:SCC<c>:PCID

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
phs_layer_cell_id: integer Range: 0 to 503

set(*phs_layer_cell_id: int*, *carrierComponent=CarrierComponent.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:CC<Nr>:PLCid
driver.configure.multiEval.cc.plcId.set(phs_layer_cell_id = 1, carrierComponent␣
↪= repcap.CarrierComponent.Default)
```

Specifies the physical layer cell ID of component carrier CC<no>. Without carrier aggregation, you can omit <no>.

INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:LTE:SIGN<i>:CELL[:PCC]:PCID
- CONFigure:LTE:SIGN<i>:CELL:SCC<c>:PCID

param phs_layer_cell_id
integer Range: 0 to 503

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

6.1.4.3 Limit

class LimitCls

Limit commands group definition. 40 total commands, 5 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.1.4.3.1 Aclr

class AclrCls

Aclr commands group definition. 8 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.1.4.3.1.1 Eutra

class EutraCls

Eutra 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.configure.multiEval.limit.aclr.eutra.clone()
```

Subgroups

6.1.4.3.1.2 CarrierAggregation

class CarrierAggregationCls

CarrierAggregation commands group definition. 3 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.1.4.3.1.3 ChannelBw1st<FirstChannelBw>

RepCap Settings

```
# Range: Bw100 .. Bw200
rc = driver.configure.multiEval.limit.aclr.eutra.carrierAggregation.channelBw1st.repcap_
↪firstChannelBw_get()
driver.configure.multiEval.limit.aclr.eutra.carrierAggregation.channelBw1st.repcap_
↪firstChannelBw_set(repcap.FirstChannelBw.Bw100)
```

class ChannelBw1stCls

ChannelBw1st commands group definition. 2 total commands, 1 Subgroups, 0 group commands Repeated Capability: FirstChannelBw, default value after init: FirstChannelBw.Bw100

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.aclr.eutra.carrierAggregation.channelBw1st.
↪clone()
```

Subgroups

6.1.4.3.1.4 ChannelBw2nd<SecondChannelBw>

RepCap Settings

```
# Range: Bw50 .. Bw200
rc = driver.configure.multiEval.limit.aclr.eutra.carrierAggregation.channelBw1st.
↪channelBw2nd.repcap_secondChannelBw_get()
driver.configure.multiEval.limit.aclr.eutra.carrierAggregation.channelBw1st.channelBw2nd.
↪repcap_secondChannelBw_set(repcap.SecondChannelBw.Bw50)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:ACLR:EUTRa:CAGGregation:CBANdwidth
↪<Band1>:CBANdwidth<Band2>
```

class ChannelBw2ndCls

ChannelBw2nd commands group definition. 2 total commands, 1 Subgroups, 1 group commands Repeated Capability: SecondChannelBw, default value after init: SecondChannelBw.Bw50

class ChannelBw2ndStruct

Response structure. Fields:

- Relative_Level: float or bool: numeric | ON | OFF Range: -256 dB to 256 dB, Unit: dB ON | OFF enables or disables the limit check.

- Absolute_Level: float or bool: numeric | ON | OFF Range: -256 dBm to 256 dBm, Unit: dBm ON | OFF enables or disables the limit check.

get(*firstChannelBw=FirstChannelBw.Default*, *secondChannelBw=SecondChannelBw.Default*) → ChannelBw2ndStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:ACLR:EUTRa:CAGGregation:CBANdwidth<Band1>:CBANdwidth<Band2>
value: ChannelBw2ndStruct = driver.configure.multiEval.limit.aclr.eutra.
↪carrierAggregation.channelBw1st.channelBw2nd.get(firstChannelBw = repcap.
↪FirstChannelBw.Default, secondChannelBw = repcap.SecondChannelBw.Default)
```

Defines relative and absolute limits for the ACLR measured in an adjacent E-UTRA channel. The settings are defined separately for each channel bandwidth combination, for two aggregated carriers. The following bandwidth combinations are supported: Example: For the first line in the figure, use …:CBANdwidth100:CBANdwidth50.

param firstChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw1st')

param secondChannelBw
optional repeated capability selector. Default value: Bw50 (settable in the interface 'ChannelBw2nd')

return
structure: for return value, see the help for ChannelBw2ndStruct structure arguments.

set(*relative_level: float*, *absolute_level: float*, *firstChannelBw=FirstChannelBw.Default*, *secondChannelBw=SecondChannelBw.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:ACLR:EUTRa:CAGGregation:CBANdwidth<Band1>:CBANdwidth<Band2>
driver.configure.multiEval.limit.aclr.eutra.carrierAggregation.channelBw1st.
↪channelBw2nd.set(relative_level = 1.0, absolute_level = 1.0, firstChannelBw =␣
↪repcap.FirstChannelBw.Default, secondChannelBw = repcap.SecondChannelBw.
↪Default)
```

Defines relative and absolute limits for the ACLR measured in an adjacent E-UTRA channel. The settings are defined separately for each channel bandwidth combination, for two aggregated carriers. The following bandwidth combinations are supported: Example: For the first line in the figure, use …:CBANdwidth100:CBANdwidth50.

param relative_level
(float or boolean) numeric | ON | OFF Range: -256 dB to 256 dB, Unit: dB ON | OFF enables or disables the limit check.

param absolute_level
(float or boolean) numeric | ON | OFF Range: -256 dBm to 256 dBm, Unit: dBm ON | OFF enables or disables the limit check.

param firstChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw1st')

param secondChannelBw
optional repeated capability selector. Default value: Bw50 (settable in the interface 'ChannelBw2nd')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.aclr.eutra.carrierAggregation.channelBw1st.
↪channelBw2nd.clone()
```

Subgroups

6.1.4.3.1.5 ChannelBw3rd<ThirdChannelBw>

RepCap Settings

```
# Range: Bw100 .. Bw200
rc = driver.configure.multiEval.limit.aclr.eutra.carrierAggregation.channelBw1st.
↪channelBw2nd.channelBw3rd.repcap_thirdChannelBw_get()
driver.configure.multiEval.limit.aclr.eutra.carrierAggregation.channelBw1st.channelBw2nd.
↪channelBw3rd.repcap_thirdChannelBw_set(repcap.ThirdChannelBw.Bw100)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:ACLR:EUTRa:CAGGregation:CBANdwidth
↪<Band1>:CBANdwidth<Band2>:CBANdwidth<Band3>
```

class ChannelBw3rdCls

ChannelBw3rd commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: ThirdChannelBw, default value after init: ThirdChannelBw.Bw100

class ChannelBw3rdStruct

Response structure. Fields:

- Relative_Level: float or bool: numeric | ON | OFF Range: -256 dB to 256 dB, Unit: dB ON | OFF enables or disables the limit check.
- Absolute_Level: float or bool: numeric | ON | OFF Range: -256 dBm to 256 dBm, Unit: dBm ON | OFF enables or disables the limit check.

get(*firstChannelBw=FirstChannelBw.Default*, *secondChannelBw=SecondChannelBw.Default*, *thirdChannelBw=ThirdChannelBw.Default*) → ChannelBw3rdStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:ACLR:EUTRa:CAGGregation:CBANdwidth<Band1>:CBANdwidth<Band2>
↪:CBANdwidth<Band3>
value: ChannelBw3rdStruct = driver.configure.multiEval.limit.aclr.eutra.
↪carrierAggregation.channelBw1st.channelBw2nd.channelBw3rd.get(firstChannelBw␣
↪= repcap.FirstChannelBw.Default, secondChannelBw = repcap.SecondChannelBw.
↪Default, thirdChannelBw = repcap.ThirdChannelBw.Default)
```

Defines relative and absolute limits for the ACLR measured in an adjacent E-UTRA channel. The settings are defined separately for each channel bandwidth combination, for three aggregated carriers. The following bandwidth combinations are supported: Example: For the first line in the figure, use ...:CBANdwidth200:CBANdwidth150:CBANdwidth100.

param firstChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw1st')

param secondChannelBw
optional repeated capability selector. Default value: Bw50 (settable in the interface 'ChannelBw2nd')

param thirdChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw3rd')

return
structure: for return value, see the help for ChannelBw3rdStruct structure arguments.

set(*relative_level: float*, *absolute_level: float*, *firstChannelBw=FirstChannelBw.Default*, *secondChannelBw=SecondChannelBw.Default*, *thirdChannelBw=ThirdChannelBw.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:ACLR:EUTRa:CAGGregation:CBANdwidth<Band1>:CBANdwidth<Band2>
↪:CBANdwidth<Band3>
driver.configure.multiEval.limit.aclr.eutra.carrierAggregation.channelBw1st.
↪channelBw2nd.channelBw3rd.set(relative_level = 1.0, absolute_level = 1.0,␣
↪firstChannelBw = repcap.FirstChannelBw.Default, secondChannelBw = repcap.
↪SecondChannelBw.Default, thirdChannelBw = repcap.ThirdChannelBw.Default)
```

Defines relative and absolute limits for the ACLR measured in an adjacent E-UTRA channel. The settings are defined separately for each channel bandwidth combination, for three aggregated carriers. The following bandwidth combinations are supported: Example: For the first line in the figure, use …:CBANdwidth200:CBANdwidth150:CBANdwidth100.

param relative_level
(float or boolean) numeric | ON | OFF Range: -256 dB to 256 dB, Unit: dB ON | OFF enables or disables the limit check.

param absolute_level
(float or boolean) numeric | ON | OFF Range: -256 dBm to 256 dBm, Unit: dBm ON | OFF enables or disables the limit check.

param firstChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw1st')

param secondChannelBw
optional repeated capability selector. Default value: Bw50 (settable in the interface 'ChannelBw2nd')

param thirdChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw3rd')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.aclr.eutra.carrierAggregation.channelBw1st.
↪channelBw2nd.channelBw3rd.clone()
```

6.1.4.3.1.6 Ocombination

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:ACLR:EUTRa:CAGGregation:OCOMbination
```

class OcombinationCls

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

class OcombinationStruct

Response structure. Fields:

- Relative_Level: float or bool: numeric | ON | OFF Range: -256 dB to 256 dB, Unit: dB ON | OFF enables or disables the limit check.
- Absolute_Level: float or bool: numeric | ON | OFF Range: -256 dBm to 256 dBm, Unit: dBm ON | OFF enables or disables the limit check.

get() → OcombinationStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:ACLR:EUTRa:CAGGregation:OCOMbination
value: OcombinationStruct = driver.configure.multiEval.limit.aclr.eutra.
↪carrierAggregation.ocombination.get()
```

Defines relative and absolute limits for the ACLR measured in an adjacent E-UTRA channel. The settings apply to all 'other' channel bandwidth combinations, not covered by other commands in this chapter.

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

set(*relative_level: float*, *absolute_level: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:ACLR:EUTRa:CAGGregation:OCOMbination
driver.configure.multiEval.limit.aclr.eutra.carrierAggregation.ocombination.
↪set(relative_level = 1.0, absolute_level = 1.0)
```

Defines relative and absolute limits for the ACLR measured in an adjacent E-UTRA channel. The settings apply to all 'other' channel bandwidth combinations, not covered by other commands in this chapter.

param relative_level
(float or boolean) numeric | ON | OFF Range: -256 dB to 256 dB, Unit: dB ON | OFF enables or disables the limit check.

param absolute_level
(float or boolean) numeric | ON | OFF Range: -256 dBm to 256 dBm, Unit: dBm ON | OFF enables or disables the limit check.

6.1.4.3.1.7 ChannelBw<ChannelBw>

RepCap Settings

```
# Range: Bw14 .. Bw200
rc = driver.configure.multiEval.limit.aclr.eutra.channelBw.repcap_channelBw_get()
driver.configure.multiEval.limit.aclr.eutra.channelBw.repcap_channelBw_set(repcap.
↪ChannelBw.Bw14)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:ACLR:EUTRa:CBANdwidth<Band>
```

class ChannelBwCls

ChannelBw commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: ChannelBw, default value after init: ChannelBw.Bw14

class ChannelBwStruct

Response structure. Fields:

- Relative_Level: float or bool: numeric | ON | OFF Range: -256 dB to 256 dB, Unit: dB ON | OFF enables or disables the limit check.
- Absolute_Level: float or bool: numeric | ON | OFF Range: -256 dBm to 256 dBm, Unit: dBm ON | OFF enables or disables the limit check.

get(*channelBw=ChannelBw.Default*) → ChannelBwStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:ACLR:EUTRa:CBANdwidth<Band>
value: ChannelBwStruct = driver.configure.multiEval.limit.aclr.eutra.channelBw.
↪get(channelBw = repcap.ChannelBw.Default)
```

Defines relative and absolute limits for the ACLR measured in an adjacent E-UTRA channel. The settings are defined separately for each channel bandwidth.

param channelBw
optional repeated capability selector. Default value: Bw14 (settable in the interface 'ChannelBw')

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

set(*relative_level: float*, *absolute_level: float*, *channelBw=ChannelBw.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:ACLR:EUTRa:CBANdwidth<Band>
driver.configure.multiEval.limit.aclr.eutra.channelBw.set(relative_level = 1.0,␣
↪absolute_level = 1.0, channelBw = repcap.ChannelBw.Default)
```

Defines relative and absolute limits for the ACLR measured in an adjacent E-UTRA channel. The settings are defined separately for each channel bandwidth.

param relative_level
(float or boolean) numeric | ON | OFF Range: -256 dB to 256 dB, Unit: dB ON | OFF enables or disables the limit check.

param absolute_level
(float or boolean) numeric | ON | OFF Range: -256 dBm to 256 dBm, Unit: dBm ON | OFF enables or disables the limit check.

param channelBw
optional repeated capability selector. Default value: Bw14 (settable in the interface 'ChannelBw')

Cloning the Group

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

6.1.4.3.1.8 Utra<UtraAdjChannel>

RepCap Settings

```
# Range: Ch1 .. Ch2
rc = driver.configure.multiEval.limit.aclr.utra.repcap_utraAdjChannel_get()
driver.configure.multiEval.limit.aclr.utra.repcap_utraAdjChannel_set(repcap.
↪UtraAdjChannel.Ch1)
```

class UtraCls
Utra commands group definition. 4 total commands, 2 Subgroups, 0 group commands Repeated Capability: UtraAdjChannel, default value after init: UtraAdjChannel.Ch1

Cloning the Group

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

Subgroups

6.1.4.3.1.9 CarrierAggregation

class CarrierAggregationCls
CarrierAggregation commands group definition. 3 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.1.4.3.1.10 ChannelBw1st<FirstChannelBw>

RepCap Settings

```
# Range: Bw100 .. Bw200
rc = driver.configure.multiEval.limit.aclr.utra.carrierAggregation.channelBw1st.repcap_
↪firstChannelBw_get()
driver.configure.multiEval.limit.aclr.utra.carrierAggregation.channelBw1st.repcap_
↪firstChannelBw_set(repcap.FirstChannelBw.Bw100)
```

class ChannelBw1stCls

ChannelBw1st commands group definition. 2 total commands, 1 Subgroups, 0 group commands Repeated Capability: FirstChannelBw, default value after init: FirstChannelBw.Bw100

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.aclr.utra.carrierAggregation.channelBw1st.
↪clone()
```

Subgroups

6.1.4.3.1.11 ChannelBw2nd<SecondChannelBw>

RepCap Settings

```
# Range: Bw50 .. Bw200
rc = driver.configure.multiEval.limit.aclr.utra.carrierAggregation.channelBw1st.
↪channelBw2nd.repcap_secondChannelBw_get()
driver.configure.multiEval.limit.aclr.utra.carrierAggregation.channelBw1st.channelBw2nd.
↪repcap_secondChannelBw_set(repcap.SecondChannelBw.Bw50)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:ACLR:UTRA<nr>
↪:CAGGregation:CBANdwidth<Band1>:CBANdwidth<Band2>
```

class ChannelBw2ndCls

ChannelBw2nd commands group definition. 2 total commands, 1 Subgroups, 1 group commands Repeated Capability: SecondChannelBw, default value after init: SecondChannelBw.Bw50

class ChannelBw2ndStruct

Response structure. Fields:

- Relative_Level: float or bool: numeric | ON | OFF Range: -256 dB to 256 dB, Unit: dB ON | OFF enables or disables the limit check.

- Absolute_Level: float or bool: numeric | ON | OFF Range: -256 dBm to 256 dBm, Unit: dBm ON | OFF enables or disables the limit check.

get(*utraAdjChannel=UtraAdjChannel.Default*, *firstChannelBw=FirstChannelBw.Default*, *secondChannelBw=SecondChannelBw.Default*) → ChannelBw2ndStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:ACLR:UTRA<nr>
↪:CAGGregation:CBANdwidth<Band1>:CBANdwidth<Band2>
value: ChannelBw2ndStruct = driver.configure.multiEval.limit.aclr.utra.
↪carrierAggregation.channelBw1st.channelBw2nd.get(utraAdjChannel = repcap.
↪UtraAdjChannel.Default, firstChannelBw = repcap.FirstChannelBw.Default,␣
↪secondChannelBw = repcap.SecondChannelBw.Default)
```

Defines relative and absolute limits for the ACLR measured in the first or second adjacent UTRA channel, depending on <no>. The settings are defined separately for each channel bandwidth combination, for two aggregated carriers. The following bandwidth combinations are supported: Example: For the first line in the figure, use ... :CBANdwidth100:CBANdwidth50.

param utraAdjChannel
optional repeated capability selector. Default value: Ch1 (settable in the interface 'Utra')

param firstChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw1st')

param secondChannelBw
optional repeated capability selector. Default value: Bw50 (settable in the interface 'ChannelBw2nd')

return
structure: for return value, see the help for ChannelBw2ndStruct structure arguments.

set(*relative_level: float*, *absolute_level: float*, *utraAdjChannel=UtraAdjChannel.Default*, *firstChannelBw=FirstChannelBw.Default*, *secondChannelBw=SecondChannelBw.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:ACLR:UTRA<nr>
↪:CAGGregation:CBANdwidth<Band1>:CBANdwidth<Band2>
driver.configure.multiEval.limit.aclr.utra.carrierAggregation.channelBw1st.
↪channelBw2nd.set(relative_level = 1.0, absolute_level = 1.0, utraAdjChannel =␣
↪repcap.UtraAdjChannel.Default, firstChannelBw = repcap.FirstChannelBw.Default,
↪ secondChannelBw = repcap.SecondChannelBw.Default)
```

Defines relative and absolute limits for the ACLR measured in the first or second adjacent UTRA channel, depending on <no>. The settings are defined separately for each channel bandwidth combination, for two aggregated carriers. The following bandwidth combinations are supported: Example: For the first line in the figure, use ... :CBANdwidth100:CBANdwidth50.

param relative_level
(float or boolean) numeric | ON | OFF Range: -256 dB to 256 dB, Unit: dB ON | OFF enables or disables the limit check.

param absolute_level
(float or boolean) numeric | ON | OFF Range: -256 dBm to 256 dBm, Unit: dBm ON | OFF enables or disables the limit check.

param utraAdjChannel
optional repeated capability selector. Default value: Ch1 (settable in the interface

'Utra')

param firstChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw1st')

param secondChannelBw
optional repeated capability selector. Default value: Bw50 (settable in the interface 'ChannelBw2nd')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.aclr.utra.carrierAggregation.channelBw1st.
↪channelBw2nd.clone()
```

Subgroups

6.1.4.3.1.12 ChannelBw3rd<ThirdChannelBw>

RepCap Settings

```
# Range: Bw100 .. Bw200
rc = driver.configure.multiEval.limit.aclr.utra.carrierAggregation.channelBw1st.
↪channelBw2nd.channelBw3rd.repcap_thirdChannelBw_get()
driver.configure.multiEval.limit.aclr.utra.carrierAggregation.channelBw1st.channelBw2nd.
↪channelBw3rd.repcap_thirdChannelBw_set(repcap.ThirdChannelBw.Bw100)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:ACLR:UTRA<nr>
↪:CAGGregation:CBANdwidth<Band1>:CBANdwidth<Band2>:CBANdwidth<Band3>
```

class ChannelBw3rdCls

ChannelBw3rd commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: ThirdChannelBw, default value after init: ThirdChannelBw.Bw100

class ChannelBw3rdStruct

Response structure. Fields:

- Relative_Level: float or bool: numeric | ON | OFF Range: -256 dB to 256 dB, Unit: dB ON | OFF enables or disables the limit check.
- Absolute_Level: float or bool: numeric | ON | OFF Range: -256 dBm to 256 dBm, Unit: dBm ON | OFF enables or disables the limit check.

get(*utraAdjChannel=UtraAdjChannel.Default*, *firstChannelBw=FirstChannelBw.Default*, *secondChannelBw=SecondChannelBw.Default*, *thirdChannelBw=ThirdChannelBw.Default*) → ChannelBw3rdStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:ACLR:UTRA<nr>
↪:CAGGregation:CBANdwidth<Band1>:CBANdwidth<Band2>:CBANdwidth<Band3>
value: ChannelBw3rdStruct = driver.configure.multiEval.limit.aclr.utra.
↪carrierAggregation.channelBw1st.channelBw2nd.channelBw3rd.get(utraAdjChannel␣
↪= repcap.UtraAdjChannel.Default, firstChannelBw = repcap.FirstChannelBw.
↪Default, secondChannelBw = repcap.SecondChannelBw.Default, thirdChannelBw =␣
↪repcap.ThirdChannelBw.Default)
```

Defines relative and absolute limits for the ACLR measured in the first or second adjacent UTRA channel, depending on <no>. The settings are defined separately for each channel bandwidth combination, for three aggregated carriers. The following bandwidth combinations are supported: Example: For the first line in the figure, use … :CBANdwidth200:CBANdwidth150:CBANdwidth100.

param utraAdjChannel
optional repeated capability selector. Default value: Ch1 (settable in the interface 'Utra')

param firstChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw1st')

param secondChannelBw
optional repeated capability selector. Default value: Bw50 (settable in the interface 'ChannelBw2nd')

param thirdChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw3rd')

return
structure: for return value, see the help for ChannelBw3rdStruct structure arguments.

set(*relative_level: float*, *absolute_level: float*, *utraAdjChannel=UtraAdjChannel.Default*, *firstChannelBw=FirstChannelBw.Default*, *secondChannelBw=SecondChannelBw.Default*, *thirdChannelBw=ThirdChannelBw.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:ACLR:UTRA<nr>
↪:CAGGregation:CBANdwidth<Band1>:CBANdwidth<Band2>:CBANdwidth<Band3>
driver.configure.multiEval.limit.aclr.utra.carrierAggregation.channelBw1st.
↪channelBw2nd.channelBw3rd.set(relative_level = 1.0, absolute_level = 1.0,␣
↪utraAdjChannel = repcap.UtraAdjChannel.Default, firstChannelBw = repcap.
↪FirstChannelBw.Default, secondChannelBw = repcap.SecondChannelBw.Default,␣
↪thirdChannelBw = repcap.ThirdChannelBw.Default)
```

Defines relative and absolute limits for the ACLR measured in the first or second adjacent UTRA channel, depending on <no>. The settings are defined separately for each channel bandwidth combination, for three aggregated carriers. The following bandwidth combinations are supported: Example: For the first line in the figure, use … :CBANdwidth200:CBANdwidth150:CBANdwidth100.

param relative_level
(float or boolean) numeric | ON | OFF Range: -256 dB to 256 dB, Unit: dB ON | OFF enables or disables the limit check.

param absolute_level
(float or boolean) numeric | ON | OFF Range: -256 dBm to 256 dBm, Unit: dBm ON | OFF enables or disables the limit check.

param utraAdjChannel
optional repeated capability selector. Default value: Ch1 (settable in the interface 'Utra')

param firstChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw1st')

param secondChannelBw
optional repeated capability selector. Default value: Bw50 (settable in the interface 'ChannelBw2nd')

param thirdChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw3rd')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.aclr.utra.carrierAggregation.channelBw1st.
↪channelBw2nd.channelBw3rd.clone()
```

6.1.4.3.1.13 Ocombination

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:ACLR:UTRA<nr>
↪:CAGGregation:OCOMbination
```

class OcombinationCls

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

class OcombinationStruct

Response structure. Fields:

- Relative_Level: float or bool: numeric | ON | OFF Range: -256 dB to 256 dB, Unit: dB ON | OFF enables or disables the limit check.
- Absolute_Level: float or bool: numeric | ON | OFF Range: -256 dBm to 256 dBm, Unit: dBm ON | OFF enables or disables the limit check.

get(*utraAdjChannel=UtraAdjChannel.Default*) → OcombinationStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:ACLR:UTRA<nr>
↪:CAGGregation:OCOMbination
value: OcombinationStruct = driver.configure.multiEval.limit.aclr.utra.
↪carrierAggregation.ocombination.get(utraAdjChannel = repcap.UtraAdjChannel.
↪Default)
```

Defines relative and absolute limits for the ACLR measured in the first or second adjacent UTRA channel, depending on <no>. The settings apply to all 'other' channel bandwidth combinations, not covered by other commands in this chapter.

param utraAdjChannel
optional repeated capability selector. Default value: Ch1 (settable in the interface 'Utra')

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

set(*relative_level: float*, *absolute_level: float*, *utraAdjChannel=UtraAdjChannel.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:ACLR:UTRA<nr>
↪:CAGGregation:OCOMbination
driver.configure.multiEval.limit.aclr.utra.carrierAggregation.ocombination.
↪set(relative_level = 1.0, absolute_level = 1.0, utraAdjChannel = repcap.
↪UtraAdjChannel.Default)
```

Defines relative and absolute limits for the ACLR measured in the first or second adjacent UTRA channel, depending on <no>. The settings apply to all 'other' channel bandwidth combinations, not covered by other commands in this chapter.

param relative_level
(float or boolean) numeric | ON | OFF Range: -256 dB to 256 dB, Unit: dB ON | OFF enables or disables the limit check.

param absolute_level
(float or boolean) numeric | ON | OFF Range: -256 dBm to 256 dBm, Unit: dBm ON | OFF enables or disables the limit check.

param utraAdjChannel
optional repeated capability selector. Default value: Ch1 (settable in the interface 'Utra')

6.1.4.3.1.14 ChannelBw<ChannelBw>

RepCap Settings

```
# Range: Bw14 .. Bw200
rc = driver.configure.multiEval.limit.aclr.utra.channelBw.repcap_channelBw_get()
driver.configure.multiEval.limit.aclr.utra.channelBw.repcap_channelBw_set(repcap.
↪ChannelBw.Bw14)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:ACLR:UTRA<nr>:CBANdwidth<Band>
```

class ChannelBwCls

ChannelBw commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: ChannelBw, default value after init: ChannelBw.Bw14

class ChannelBwStruct

Response structure. Fields:

- Relative_Level: float or bool: numeric | ON | OFF Range: -256 dB to 256 dB, Unit: dB ON | OFF enables or disables the limit check.

- Absolute_Level: float or bool: numeric | ON | OFF Range: -256 dBm to 256 dBm, Unit: dBm ON | OFF enables or disables the limit check.

get(*utraAdjChannel=UtraAdjChannel.Default*, *channelBw=ChannelBw.Default*) → ChannelBwStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:ACLR:UTRA<nr>
↪:CBANdwidth<Band>
value: ChannelBwStruct = driver.configure.multiEval.limit.aclr.utra.channelBw.
↪get(utraAdjChannel = repcap.UtraAdjChannel.Default, channelBw = repcap.
↪ChannelBw.Default)
```

Defines relative and absolute limits for the ACLR measured in the first or second adjacent UTRA channel, depending on <no>. The settings are defined separately for each channel bandwidth.

param utraAdjChannel
optional repeated capability selector. Default value: Ch1 (settable in the interface 'Utra')

param channelBw
optional repeated capability selector. Default value: Bw14 (settable in the interface 'ChannelBw')

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

set(*relative_level: float*, *absolute_level: float*, *utraAdjChannel=UtraAdjChannel.Default*, *channelBw=ChannelBw.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:ACLR:UTRA<nr>
↪:CBANdwidth<Band>
driver.configure.multiEval.limit.aclr.utra.channelBw.set(relative_level = 1.0,␣
↪absolute_level = 1.0, utraAdjChannel = repcap.UtraAdjChannel.Default,␣
↪channelBw = repcap.ChannelBw.Default)
```

Defines relative and absolute limits for the ACLR measured in the first or second adjacent UTRA channel, depending on <no>. The settings are defined separately for each channel bandwidth.

param relative_level
(float or boolean) numeric | ON | OFF Range: -256 dB to 256 dB, Unit: dB ON | OFF enables or disables the limit check.

param absolute_level
(float or boolean) numeric | ON | OFF Range: -256 dBm to 256 dBm, Unit: dBm ON | OFF enables or disables the limit check.

param utraAdjChannel
optional repeated capability selector. Default value: Ch1 (settable in the interface 'Utra')

param channelBw
optional repeated capability selector. Default value: Bw14 (settable in the interface 'ChannelBw')

Cloning the Group

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

6.1.4.3.2 Pdynamics

class PdynamicsCls

Pdynamics 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.multiEval.limit.pdynamics.clone()
```

Subgroups

6.1.4.3.2.1 ChannelBw<ChannelBw>

RepCap Settings

```
# Range: Bw14 .. Bw200
rc = driver.configure.multiEval.limit.pdynamics.channelBw.repcap_channelBw_get()
driver.configure.multiEval.limit.pdynamics.channelBw.repcap_channelBw_set(repcap.
↪ChannelBw.Bw14)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:PDYNamics:CBANdwidth<Band>
```

class ChannelBwCls

ChannelBw commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: ChannelBw, default value after init: ChannelBw.Bw14

class ChannelBwStruct

Response structure. Fields:

- Enable: bool: OFF | ON OFF: disables the limit check ON: enables the limit check
- On_Power_Upper: float: numeric Upper limit for the 'ON power' Range: -256 dBm to 256 dBm, Unit: dBm
- On_Power_Lower: float: numeric Lower limit for the 'ON power' Range: -256 dBm to 256 dBm, Unit: dBm
- Off_Power_Upper: float: numeric Upper limit for the 'OFF power' and the 'SRS OFF' power Range: -256 dBm to 256 dBm, Unit: dBm

get(*channelBw=ChannelBw.Default*) → ChannelBwStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:PDYNamics:CBANdwidth<Band>
value: ChannelBwStruct = driver.configure.multiEval.limit.pdynamics.channelBw.
↪get(channelBw = repcap.ChannelBw.Default)
```

Defines limits for the ON power and OFF power determined with the power dynamics measurement. Separate limits can be defined for each channel bandwidth.

param channelBw
optional repeated capability selector. Default value: Bw14 (settable in the interface 'ChannelBw')

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

set(*enable: bool*, *on_power_upper: float*, *on_power_lower: float*, *off_power_upper: float*, *channelBw=ChannelBw.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:PDYNamics:CBANdwidth<Band>
driver.configure.multiEval.limit.pdynamics.channelBw.set(enable = False, on_
↪power_upper = 1.0, on_power_lower = 1.0, off_power_upper = 1.0, channelBw =␣
↪repcap.ChannelBw.Default)
```

Defines limits for the ON power and OFF power determined with the power dynamics measurement. Separate limits can be defined for each channel bandwidth.

param enable
OFF | ON OFF: disables the limit check ON: enables the limit check

param on_power_upper
numeric Upper limit for the 'ON power' Range: -256 dBm to 256 dBm, Unit: dBm

param on_power_lower
numeric Lower limit for the 'ON power' Range: -256 dBm to 256 dBm, Unit: dBm

param off_power_upper
numeric Upper limit for the 'OFF power' and the 'SRS OFF' power Range: -256 dBm to 256 dBm, Unit: dBm

param channelBw
optional repeated capability selector. Default value: Bw14 (settable in the interface 'ChannelBw')

Cloning the Group

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

6.1.4.3.3 Qam<QAMmodOrder>

RepCap Settings

```
# Range: Qam16 .. Qam256
rc = driver.configure.multiEval.limit.qam.repcap_qAMmodOrder_get()
driver.configure.multiEval.limit.qam.repcap_qAMmodOrder_set(repcap.QAMmodOrder.Qam16)
```

class QamCls

Qam commands group definition. 9 total commands, 8 Subgroups, 0 group commands Repeated Capability: QAMmodOrder, default value after init: QAMmodOrder.Qam16

Cloning the Group

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

Subgroups

6.1.4.3.3.1 EsFlatness

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>:ESFLatness
```

class EsFlatnessCls

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

class EsFlatnessStruct

Structure for setting input parameters. Fields:

- Enable: bool: OFF | ON OFF: disables the limit check ON: enables the limit check
- Range_1: float: numeric Upper limit for max(range 1) - min(range 1) Range: -256 dBpp to 256 dBpp, Unit: dBpp
- Range_2: float: numeric Upper limit for max(range 2) - min(range 2) Range: -256 dBpp to 256 dBpp, Unit: dBpp
- Max_1_Min_2: float: numeric Upper limit for max(range 1) - min(range 2) Range: -256 dB to 256 dB, Unit: dB
- Max_2_Min_1: float: numeric Upper limit for max(range 2) - min(range 1) Range: -256 dB to 256 dB, Unit: dB
- Edge_Frequency: float: numeric Frequency band edge distance of border between range 1 and range 2 Range: 0 MHz to 20 MHz, Unit: Hz

get(*qAMmodOrder=QAMmodOrder.Default*) → EsFlatnessStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:ESFLatness
value: EsFlatnessStruct = driver.configure.multiEval.limit.qam.esFlatness.
↪get(qAMmodOrder = repcap.QAMmodOrder.Default)
```

Defines limits for the equalizer spectrum flatness, for QAM modulations.

param qAMmodOrder
optional repeated capability selector. Default value: Qam16 (settable in the interface 'Qam')

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

set(*structure: EsFlatnessStruct*, *qAMmodOrder=QAMmodOrder.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:ESFLatness
structure = driver.configure.multiEval.limit.qam.esFlatness.EsFlatnessStruct()
structure.Enable: bool = False
structure.Range_1: float = 1.0
structure.Range_2: float = 1.0
structure.Max_1_Min_2: float = 1.0
structure.Max_2_Min_1: float = 1.0
structure.Edge_Frequency: float = 1.0
driver.configure.multiEval.limit.qam.esFlatness.set(structure, qAMmodOrder =␣
↪repcap.QAMmodOrder.Default)
```

Defines limits for the equalizer spectrum flatness, for QAM modulations.

param structure
for set value, see the help for EsFlatnessStruct structure arguments.

param qAMmodOrder
optional repeated capability selector. Default value: Qam16 (settable in the interface 'Qam')

6.1.4.3.3.2 EvMagnitude

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>:EVMagnitude
```

class EvMagnitudeCls

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

class EvMagnitudeStruct

Response structure. Fields:

- Rms: float or bool: numeric | ON | OFF Range: 0 % to 100 %, Unit: % ON | OFF enables or disables the limit check.
- Peak: float or bool: numeric | ON | OFF Range: 0 % to 100 %, Unit: % ON | OFF enables or disables the limit check.

get(*qAMmodOrder=QAMmodOrder.Default*) → EvMagnitudeStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:EVMagnitude
value: EvMagnitudeStruct = driver.configure.multiEval.limit.qam.evMagnitude.
↪get(qAMmodOrder = repcap.QAMmodOrder.Default)
```

Defines upper limits for the RMS and peak values of the error vector magnitude (EVM) , for QAM modulations.

param qAMmodOrder
optional repeated capability selector. Default value: Qam16 (settable in the interface 'Qam')

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

set(*rms: float*, *peak: float*, *qAMmodOrder=QAMmodOrder.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:EVMagnitude
driver.configure.multiEval.limit.qam.evMagnitude.set(rms = 1.0, peak = 1.0,␣
↪qAMmodOrder = repcap.QAMmodOrder.Default)
```

Defines upper limits for the RMS and peak values of the error vector magnitude (EVM) , for QAM modulations.

param rms
(float or boolean) numeric | ON | OFF Range: 0 % to 100 %, Unit: % ON | OFF enables or disables the limit check.

param peak
(float or boolean) numeric | ON | OFF Range: 0 % to 100 %, Unit: % ON | OFF enables or disables the limit check.

param qAMmodOrder
optional repeated capability selector. Default value: Qam16 (settable in the interface 'Qam')

6.1.4.3.3.3 FreqError

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>:FERRor
```

class FreqErrorCls

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

get(*qAMmodOrder=QAMmodOrder.Default*) → float

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:FERRor
value: float or bool = driver.configure.multiEval.limit.qam.freqError.
↪get(qAMmodOrder = repcap.QAMmodOrder.Default)
```

Defines an upper limit for the carrier frequency error, for QAM modulations.

param qAMmodOrder
optional repeated capability selector. Default value: Qam16 (settable in the interface 'Qam')

return
frequency_error: (float or boolean) numeric | ON | OFF Range: 0 ppm to 1 ppm, Unit: ppm ON | OFF enables or disables the limit check.

set(*frequency_error: float*, *qAMmodOrder=QAMmodOrder.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:FERRor
driver.configure.multiEval.limit.qam.freqError.set(frequency_error = 1.0,␣
↪qAMmodOrder = repcap.QAMmodOrder.Default)
```

Defines an upper limit for the carrier frequency error, for QAM modulations.

param frequency_error
(float or boolean) numeric | ON | OFF Range: 0 ppm to 1 ppm, Unit: ppm ON | OFF enables or disables the limit check.

param qAMmodOrder
optional repeated capability selector. Default value: Qam16 (settable in the interface 'Qam')

6.1.4.3.3.4 Ibe

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>:IBE
```

class IbeCls

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

class IbeStruct

Response structure. Fields:

- Enable: bool: OFF | ON OFF: disables the limit check ON: enables the limit check
- Minimum: float: numeric Range: -256 dB to 256 dB, Unit: dB
- Evm: float: numeric Range: 0 % to 100 %, Unit: %
- Rb_Power: float: numeric Range: -256 dBm to 256 dBm, Unit: dBm
- Iq_Image: float: numeric Range: -256 dB to 256 dB, Unit: dB

get(*qAMmodOrder=QAMmodOrder.Default*) → IbeStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>:IBE
value: IbeStruct = driver.configure.multiEval.limit.qam.ibe.get(qAMmodOrder =␣
↪repcap.QAMmodOrder.Default)
```

Defines parameters used for calculation of an upper limit for the inband emission, for QAM modulations, see 'Inband emissions limits'.

param qAMmodOrder
optional repeated capability selector. Default value: Qam16 (settable in the interface 'Qam')

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

set(*enable: bool*, *minimum: float*, *evm: float*, *rb_power: float*, *iq_image: float*, *qAMmodOrder=QAMmodOrder.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>:IBE
driver.configure.multiEval.limit.qam.ibe.set(enable = False, minimum = 1.0, evm␣
↪= 1.0, rb_power = 1.0, iq_image = 1.0, qAMmodOrder = repcap.QAMmodOrder.
↪Default)
```

Defines parameters used for calculation of an upper limit for the inband emission, for QAM modulations, see 'Inband emissions limits'.

param enable
OFF | ON OFF: disables the limit check ON: enables the limit check

param minimum
numeric Range: -256 dB to 256 dB, Unit: dB

param evm
numeric Range: 0 % to 100 %, Unit: %

param rb_power
numeric Range: -256 dBm to 256 dBm, Unit: dBm

param iq_image
numeric Range: -256 dB to 256 dB, Unit: dB

param qAMmodOrder
optional repeated capability selector. Default value: Qam16 (settable in the interface 'Qam')

Cloning the Group

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

Subgroups

6.1.4.3.3.5 IqOffset

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>:IBE:IQOFfset
```

class IqOffsetCls

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

class IqOffsetStruct

Response structure. Fields:

- Offset_1: float: numeric Offset for high TX power range Range: -256 dBc to 256 dBc, Unit: dBc
- Offset_2: float: numeric Offset for intermediate TX power range Range: -256 dBc to 256 dBc, Unit: dBc
- Offset_3: float: numeric Offset for low TX power range Range: -256 dBc to 256 dBc, Unit: dBc

get(*qAMmodOrder=QAMmodOrder.Default*) → IqOffsetStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:IBE:IQOFfset
value: IqOffsetStruct = driver.configure.multiEval.limit.qam.ibe.iqOffset.
↪get(qAMmodOrder = repcap.QAMmodOrder.Default)
```

Defines I/Q origin offset values used for calculation of an upper limit for the inband emission, for QAM modulations. Three different values can be set for three TX power ranges, see 'Inband emissions limits'.

param qAMmodOrder
optional repeated capability selector. Default value: Qam16 (settable in the interface 'Qam')

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

set(*offset_1: float*, *offset_2: float*, *offset_3: float*, *qAMmodOrder=QAMmodOrder.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:IBE:IQOFfset
driver.configure.multiEval.limit.qam.ibe.iqOffset.set(offset_1 = 1.0, offset_2␣
↪= 1.0, offset_3 = 1.0, qAMmodOrder = repcap.QAMmodOrder.Default)
```

Defines I/Q origin offset values used for calculation of an upper limit for the inband emission, for QAM modulations. Three different values can be set for three TX power ranges, see 'Inband emissions limits'.

param offset_1
numeric Offset for high TX power range Range: -256 dBc to 256 dBc, Unit: dBc

param offset_2
numeric Offset for intermediate TX power range Range: -256 dBc to 256 dBc, Unit: dBc

param offset_3
numeric Offset for low TX power range Range: -256 dBc to 256 dBc, Unit: dBc

param qAMmodOrder
optional repeated capability selector. Default value: Qam16 (settable in the interface 'Qam')

6.1.4.3.3.6 IqOffset

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>:IQOFfset
```

class IqOffsetCls

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

class IqOffsetStruct

Response structure. Fields:

- Enable: bool: OFF | ON OFF: disables the limit check ON: enables the limit check
- Offset_1: float: numeric I/Q origin offset limit for high TX power range Range: -256 dBc to 256 dBc, Unit: dBc

- Offset_2: float: numeric I/Q origin offset limit for intermediate TX power range Range: -256 dBc to 256 dBc, Unit: dBc
- Offset_3: float: numeric I/Q origin offset limit for low TX power range Range: -256 dBc to 256 dBc, Unit: dBc

get(*qAMmodOrder=QAMmodOrder.Default*) → IqOffsetStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:IQOFfset
value: IqOffsetStruct = driver.configure.multiEval.limit.qam.iqOffset.
↪get(qAMmodOrder = repcap.QAMmodOrder.Default)
```

Defines upper limits for the I/Q origin offset, for QAM modulations. Three different I/Q origin offset limits can be set for three TX power ranges. For details, see 'I/Q origin offset limits'.

param qAMmodOrder
optional repeated capability selector. Default value: Qam16 (settable in the interface 'Qam')

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

set(*enable: bool*, *offset_1: float*, *offset_2: float*, *offset_3: float*, *qAMmodOrder=QAMmodOrder.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:IQOFfset
driver.configure.multiEval.limit.qam.iqOffset.set(enable = False, offset_1 = 1.
↪0, offset_2 = 1.0, offset_3 = 1.0, qAMmodOrder = repcap.QAMmodOrder.Default)
```

Defines upper limits for the I/Q origin offset, for QAM modulations. Three different I/Q origin offset limits can be set for three TX power ranges. For details, see 'I/Q origin offset limits'.

param enable
OFF | ON OFF: disables the limit check ON: enables the limit check

param offset_1
numeric I/Q origin offset limit for high TX power range Range: -256 dBc to 256 dBc, Unit: dBc

param offset_2
numeric I/Q origin offset limit for intermediate TX power range Range: -256 dBc to 256 dBc, Unit: dBc

param offset_3
numeric I/Q origin offset limit for low TX power range Range: -256 dBc to 256 dBc, Unit: dBc

param qAMmodOrder
optional repeated capability selector. Default value: Qam16 (settable in the interface 'Qam')

6.1.4.3.3.7 Merror

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>:MERRor
```

class MerrorCls

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

class MerrorStruct

Response structure. Fields:

- Rms: float or bool: numeric | ON | OFF Range: 0 % to 100 %, Unit: % ON | OFF enables or disables the limit check.
- Peak: float or bool: numeric | ON | OFF Range: 0 % to 100 %, Unit: % ON | OFF enables or disables the limit check.

get(*qAMmodOrder=QAMmodOrder.Default*) → MerrorStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:MERRor
value: MerrorStruct = driver.configure.multiEval.limit.qam.merror.
↪get(qAMmodOrder = repcap.QAMmodOrder.Default)
```

Defines upper limits for the RMS and peak values of the magnitude error, for QAM modulations.

param qAMmodOrder
optional repeated capability selector. Default value: Qam16 (settable in the interface 'Qam')

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

set(*rms: float*, *peak: float*, *qAMmodOrder=QAMmodOrder.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:MERRor
driver.configure.multiEval.limit.qam.merror.set(rms = 1.0, peak = 1.0,␣
↪qAMmodOrder = repcap.QAMmodOrder.Default)
```

Defines upper limits for the RMS and peak values of the magnitude error, for QAM modulations.

param rms
(float or boolean) numeric | ON | OFF Range: 0 % to 100 %, Unit: % ON | OFF enables or disables the limit check.

param peak
(float or boolean) numeric | ON | OFF Range: 0 % to 100 %, Unit: % ON | OFF enables or disables the limit check.

param qAMmodOrder
optional repeated capability selector. Default value: Qam16 (settable in the interface 'Qam')

6.1.4.3.3.8 Perror

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>:PERRor
```

class PerrorCls

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

class PerrorStruct

Response structure. Fields:

- Rms: float or bool: numeric | ON | OFF Range: 0 deg to 180 deg, Unit: deg ON | OFF enables or disables the limit check.
- Peak: float or bool: numeric | ON | OFF Range: 0 deg to 180 deg, Unit: deg ON | OFF enables or disables the limit check.

get(*qAMmodOrder=QAMmodOrder.Default*) → PerrorStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:PERRor
value: PerrorStruct = driver.configure.multiEval.limit.qam.perror.
↪get(qAMmodOrder = repcap.QAMmodOrder.Default)
```

Defines symmetric limits for the RMS and peak values of the phase error, for QAM modulations. The limit check fails if the absolute value of the measured phase error exceeds the specified values.

param qAMmodOrder
optional repeated capability selector. Default value: Qam16 (settable in the interface 'Qam')

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

set(*rms: float*, *peak: float*, *qAMmodOrder=QAMmodOrder.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:PERRor
driver.configure.multiEval.limit.qam.perror.set(rms = 1.0, peak = 1.0,␣
↪qAMmodOrder = repcap.QAMmodOrder.Default)
```

Defines symmetric limits for the RMS and peak values of the phase error, for QAM modulations. The limit check fails if the absolute value of the measured phase error exceeds the specified values.

param rms
(float or boolean) numeric | ON | OFF Range: 0 deg to 180 deg, Unit: deg ON | OFF enables or disables the limit check.

param peak
(float or boolean) numeric | ON | OFF Range: 0 deg to 180 deg, Unit: deg ON | OFF enables or disables the limit check.

param qAMmodOrder
optional repeated capability selector. Default value: Qam16 (settable in the interface 'Qam')

6.1.4.3.3.9 Sflatness

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>:SFLatness
```

class SflatnessCls

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

class SflatnessStruct

Structure for setting input parameters. Fields:

- Enable: bool: No parameter help available
- Lower: float: No parameter help available
- Upper: float: No parameter help available
- Edge_Lower: float: No parameter help available
- Edge_Upper: float: No parameter help available
- Edge_Frequency: float: No parameter help available

get(*qAMmodOrder=QAMmodOrder.Default*) → SflatnessStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:SFLatness
value: SflatnessStruct = driver.configure.multiEval.limit.qam.sflatness.
↪get(qAMmodOrder = repcap.QAMmodOrder.Default)
```

No command help available

param qAMmodOrder
optional repeated capability selector. Default value: Qam16 (settable in the interface 'Qam')

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

set(*structure: SflatnessStruct*, *qAMmodOrder=QAMmodOrder.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:SFLatness
structure = driver.configure.multiEval.limit.qam.sflatness.SflatnessStruct()
structure.Enable: bool = False
structure.Lower: float = 1.0
structure.Upper: float = 1.0
structure.Edge_Lower: float = 1.0
structure.Edge_Upper: float = 1.0
structure.Edge_Frequency: float = 1.0
driver.configure.multiEval.limit.qam.sflatness.set(structure, qAMmodOrder =␣
↪repcap.QAMmodOrder.Default)
```

No command help available

param structure
for set value, see the help for SflatnessStruct structure arguments.

param qAMmodOrder
optional repeated capability selector. Default value: Qam16 (settable in the interface 'Qam')

6.1.4.3.4 Qpsk

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:FERRor
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:SFLatness
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:ESFLatness
```

class QpskCls

Qpsk commands group definition. 9 total commands, 5 Subgroups, 3 group commands

class EsFlatnessStruct

Structure for setting input parameters. Fields:

- Enable: bool: OFF | ON OFF: disables the limit check ON: enables the limit check
- Range_1: float: numeric Upper limit for max(range 1) - min(range 1) Range: -256 dBpp to 256 dBpp, Unit: dBpp
- Range_2: float: numeric Upper limit for max(range 2) - min(range 2) Range: -256 dBpp to 256 dBpp, Unit: dBpp
- Max_1_Min_2: float: numeric Upper limit for max(range 1) - min(range 2) Range: -256 dB to 256 dB, Unit: dB
- Max_2_Min_1: float: numeric Upper limit for max(range 2) - min(range 1) Range: -256 dB to 256 dB, Unit: dB
- Edge_Frequency: float: numeric Frequency band edge distance of border between range 1 and range 2 Range: 0 MHz to 20 MHz, Unit: Hz

class SflatnessStruct

Structure for setting input parameters. Fields:

- Enable: bool: No parameter help available
- Lower: float: No parameter help available
- Upper: float: No parameter help available
- Edge_Lower: float: No parameter help available
- Edge_Upper: float: No parameter help available
- Edge_Frequency: float: No parameter help available

get_es_flatness() → EsFlatnessStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:ESFLatness
value: EsFlatnessStruct = driver.configure.multiEval.limit.qpsk.get_es_
↪flatness()
```

Defines limits for the equalizer spectrum flatness (QPSK modulation) .

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

get_freq_error() → float

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:FERRor
value: float or bool = driver.configure.multiEval.limit.qpsk.get_freq_error()
```

Defines an upper limit for the carrier frequency error (QPSK modulation) .

return
frequency_error: (float or boolean) numeric | ON | OFF Range: 0 ppm to 1 ppm, Unit: ppm ON | OFF enables or disables the limit check.

get_sflatness() → SflatnessStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:SFLatness
value: SflatnessStruct = driver.configure.multiEval.limit.qpsk.get_sflatness()
```

No command help available

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

set_es_flatness(*value: EsFlatnessStruct*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:ESFLatness
structure = driver.configure.multiEval.limit.qpsk.EsFlatnessStruct()
structure.Enable: bool = False
structure.Range_1: float = 1.0
structure.Range_2: float = 1.0
structure.Max_1_Min_2: float = 1.0
structure.Max_2_Min_1: float = 1.0
structure.Edge_Frequency: float = 1.0
driver.configure.multiEval.limit.qpsk.set_es_flatness(value = structure)
```

Defines limits for the equalizer spectrum flatness (QPSK modulation) .

param value
see the help for EsFlatnessStruct structure arguments.

set_freq_error(*frequency_error: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:FERRor
driver.configure.multiEval.limit.qpsk.set_freq_error(frequency_error = 1.0)
```

Defines an upper limit for the carrier frequency error (QPSK modulation) .

param frequency_error
(float or boolean) numeric | ON | OFF Range: 0 ppm to 1 ppm, Unit: ppm ON | OFF enables or disables the limit check.

set_sflatness(*value: SflatnessStruct*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:SFLatness
structure = driver.configure.multiEval.limit.qpsk.SflatnessStruct()
structure.Enable: bool = False
structure.Lower: float = 1.0
structure.Upper: float = 1.0
structure.Edge_Lower: float = 1.0
```

(continues on next page)

(continued from previous page)

```
structure.Edge_Upper: float = 1.0
structure.Edge_Frequency: float = 1.0
driver.configure.multiEval.limit.qpsk.set_sflatness(value = structure)
```

No command help available

param value
see the help for SflatnessStruct structure arguments.

Cloning the Group

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

Subgroups

6.1.4.3.4.1 EvMagnitude

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:EVMagnitude
```

class EvMagnitudeCls

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

class EvMagnitudeStruct

Response structure. Fields:

- Rms: float or bool: numeric | ON | OFF Range: 0 % to 100 %, Unit: % ON | OFF enables or disables the limit check.
- Peak: float or bool: numeric | ON | OFF Range: 0 % to 100 %, Unit: % ON | OFF enables or disables the limit check.

get() → EvMagnitudeStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:EVMagnitude
value: EvMagnitudeStruct = driver.configure.multiEval.limit.qpsk.evMagnitude.
↪get()
```

Defines upper limits for the RMS and peak values of the error vector magnitude (EVM) for QPSK.

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

set(*rms: float*, *peak: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:EVMagnitude
driver.configure.multiEval.limit.qpsk.evMagnitude.set(rms = 1.0, peak = 1.0)
```

Defines upper limits for the RMS and peak values of the error vector magnitude (EVM) for QPSK.

param rms
(float or boolean) numeric | ON | OFF Range: 0 % to 100 %, Unit: % ON | OFF enables or disables the limit check.

param peak
(float or boolean) numeric | ON | OFF Range: 0 % to 100 %, Unit: % ON | OFF enables or disables the limit check.

6.1.4.3.4.2 Ibe

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:IBE
```

class IbeCls

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

class IbeStruct

Response structure. Fields:

- Enable: bool: OFF | ON OFF: disables the limit check ON: enables the limit check
- Minimum: float: numeric Range: -256 dB to 256 dB, Unit: dB
- Evm: float: numeric Range: 0 % to 100 %, Unit: %
- Rb_Power: float: numeric Range: -256 dBm to 256 dBm, Unit: dBm
- Iq_Image: float: numeric Range: -256 dB to 256 dB, Unit: dB

get() → IbeStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:IBE
value: IbeStruct = driver.configure.multiEval.limit.qpsk.ibe.get()
```

Defines parameters used for calculation of an upper limit for the inband emission (QPSK modulation) , see 'Inband emissions limits'.

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

set(*enable: bool*, *minimum: float*, *evm: float*, *rb_power: float*, *iq_image: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:IBE
driver.configure.multiEval.limit.qpsk.ibe.set(enable = False, minimum = 1.0,␣
↪evm = 1.0, rb_power = 1.0, iq_image = 1.0)
```

Defines parameters used for calculation of an upper limit for the inband emission (QPSK modulation) , see 'Inband emissions limits'.

param enable
OFF | ON OFF: disables the limit check ON: enables the limit check

param minimum
numeric Range: -256 dB to 256 dB, Unit: dB

param evm
numeric Range: 0 % to 100 %, Unit: %

param rb_power
numeric Range: -256 dBm to 256 dBm, Unit: dBm

param iq_image
numeric Range: -256 dB to 256 dB, Unit: dB

Cloning the Group

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

Subgroups

6.1.4.3.4.3 IqOffset

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:IBE:IQOFfset
```

class IqOffsetCls

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

class IqOffsetStruct

Response structure. Fields:

- Offset_1: float: numeric Offset for high TX power range Range: -256 dBc to 256 dBc, Unit: dB
- Offset_2: float: numeric Offset for intermediate TX power range Range: -256 dBc to 256 dBc, Unit: dB
- Offset_3: float: numeric Offset for low TX power range Range: -256 dBc to 256 dBc, Unit: dB

get() → IqOffsetStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:IBE:IQOFfset
value: IqOffsetStruct = driver.configure.multiEval.limit.qpsk.ibe.iqOffset.get()
```

Defines I/Q origin offset values used for calculation of an upper limit for the inband emission (QPSK modulation) . Three different values can be set for three TX power ranges, see 'Inband emissions limits'.

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

set(*offset_1: float*, *offset_2: float*, *offset_3: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:IBE:IQOFfset
driver.configure.multiEval.limit.qpsk.ibe.iqOffset.set(offset_1 = 1.0, offset_2␣
↪= 1.0, offset_3 = 1.0)
```

Defines I/Q origin offset values used for calculation of an upper limit for the inband emission (QPSK modulation) . Three different values can be set for three TX power ranges, see 'Inband emissions limits'.

param offset_1
numeric Offset for high TX power range Range: -256 dBc to 256 dBc, Unit: dB

param offset_2
numeric Offset for intermediate TX power range Range: -256 dBc to 256 dBc, Unit: dB

param offset_3
numeric Offset for low TX power range Range: -256 dBc to 256 dBc, Unit: dB

6.1.4.3.4.4 IqOffset

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:IQOFfset
```

class IqOffsetCls

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

class IqOffsetStruct

Response structure. Fields:

- Enable: bool: OFF | ON OFF: disables the limit check ON: enables the limit check
- Offset_1: float: numeric I/Q origin offset limit for high TX power range Range: -256 dBc to 256 dBc, Unit: dB
- Offset_2: float: numeric I/Q origin offset limit for intermediate TX power range Range: -256 dBc to 256 dBc, Unit: dB
- Offset_3: float: numeric I/Q origin offset limit for low TX power range Range: -256 dBc to 256 dBc, Unit: dB

get() → IqOffsetStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:IQOFfset
value: IqOffsetStruct = driver.configure.multiEval.limit.qpsk.iqOffset.get()
```

Defines upper limits for the I/Q origin offset (QPSK modulation) . Three different I/Q origin offset limits can be set for three TX power ranges. For details, see 'I/Q origin offset limits'.

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

set(*enable: bool*, *offset_1: float*, *offset_2: float*, *offset_3: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:IQOFfset
driver.configure.multiEval.limit.qpsk.iqOffset.set(enable = False, offset_1 = 1.
↪0, offset_2 = 1.0, offset_3 = 1.0)
```

Defines upper limits for the I/Q origin offset (QPSK modulation) . Three different I/Q origin offset limits can be set for three TX power ranges. For details, see 'I/Q origin offset limits'.

param enable
OFF | ON OFF: disables the limit check ON: enables the limit check

param offset_1
numeric I/Q origin offset limit for high TX power range Range: -256 dBc to 256 dBc, Unit: dB

param offset_2
numeric I/Q origin offset limit for intermediate TX power range Range: -256 dBc to 256 dBc, Unit: dB

param offset_3
numeric I/Q origin offset limit for low TX power range Range: -256 dBc to 256 dBc, Unit: dB

6.1.4.3.4.5 Merror

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:MERRor
```

class MerrorCls

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

class MerrorStruct

Response structure. Fields:

- Rms: float or bool: numeric | ON | OFF Range: 0 % to 100 %, Unit: % ON | OFF enables or disables the limit check.
- Peak: float or bool: numeric | ON | OFF Range: 0 % to 100 %, Unit: % ON | OFF enables or disables the limit check.

get() → MerrorStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:MERRor
value: MerrorStruct = driver.configure.multiEval.limit.qpsk.merror.get()
```

Defines upper limits for the RMS and peak values of the magnitude error for QPSK.

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

set(*rms: float*, *peak: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:MERRor
driver.configure.multiEval.limit.qpsk.merror.set(rms = 1.0, peak = 1.0)
```

Defines upper limits for the RMS and peak values of the magnitude error for QPSK.

param rms
(float or boolean) numeric | ON | OFF Range: 0 % to 100 %, Unit: % ON | OFF enables or disables the limit check.

param peak
(float or boolean) numeric | ON | OFF Range: 0 % to 100 %, Unit: % ON | OFF enables or disables the limit check.

6.1.4.3.4.6 Perror

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:PERRor
```

class PerrorCls

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

class PerrorStruct

Response structure. Fields:

- Rms: float or bool: numeric | ON | OFF Range: 0 deg to 180 deg, Unit: deg ON | OFF enables or disables the limit check.
- Peak: float or bool: numeric | ON | OFF Range: 0 deg to 180 deg, Unit: deg ON | OFF enables or disables the limit check.

get() → PerrorStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:PERRor
value: PerrorStruct = driver.configure.multiEval.limit.qpsk.perror.get()
```

Defines symmetric limits for the RMS and peak values of the phase error for QPSK. The limit check fails if the absolute value of the measured phase error exceeds the specified values.

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

set(*rms: float*, *peak: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:PERRor
driver.configure.multiEval.limit.qpsk.perror.set(rms = 1.0, peak = 1.0)
```

Defines symmetric limits for the RMS and peak values of the phase error for QPSK. The limit check fails if the absolute value of the measured phase error exceeds the specified values.

param rms
(float or boolean) numeric | ON | OFF Range: 0 deg to 180 deg, Unit: deg ON | OFF enables or disables the limit check.

param peak
(float or boolean) numeric | ON | OFF Range: 0 deg to 180 deg, Unit: deg ON | OFF enables or disables the limit check.

6.1.4.3.5 SeMask

class SeMaskCls

SeMask commands group definition. 13 total commands, 3 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.1.4.3.5.1 AtTolerance<EutraBand>

RepCap Settings

```
# Range: Nr30 .. Nr50
rc = driver.configure.multiEval.limit.seMask.atTolerance.repcap_eutraBand_get()
driver.configure.multiEval.limit.seMask.atTolerance.repcap_eutraBand_set(repcap.
↪EutraBand.Nr30)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:ATTolerance<EUTRAband>
```

class AtToleranceCls

AtTolerance commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: EutraBand, default value after init: EutraBand.Nr30

get(*eutraBand=EutraBand.Default*) → float

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:ATTolerance
↪<EUTRAband>
value: float = driver.configure.multiEval.limit.seMask.atTolerance.
↪get(eutraBand = repcap.EutraBand.Default)
```

Defines additional test tolerances for the emission masks. The tolerance is added to the power values of all general and additional spectrum emission masks. A positive tolerance value relaxes the limits. For operating bands below 3 GHz, there is no additional test tolerance. You can define different additional test tolerances for bands above 3 GHz and for bands above 5 GHz.

param eutraBand
optional repeated capability selector. Default value: Nr30 (settable in the interface 'AtTolerance')

return
add_test_tol: numeric Additional test tolerance Range: -5 dB to 5 dB, Unit: dB

set(*add_test_tol: float*, *eutraBand=EutraBand.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:ATTolerance
↪<EUTRAband>
driver.configure.multiEval.limit.seMask.atTolerance.set(add_test_tol = 1.0,␣
↪eutraBand = repcap.EutraBand.Default)
```

Defines additional test tolerances for the emission masks. The tolerance is added to the power values of all general and additional spectrum emission masks. A positive tolerance value relaxes the limits. For

operating bands below 3 GHz, there is no additional test tolerance. You can define different additional test tolerances for bands above 3 GHz and for bands above 5 GHz.

param add_test_tol
numeric Additional test tolerance Range: -5 dB to 5 dB, Unit: dB

param eutraBand
optional repeated capability selector. Default value: Nr30 (settable in the interface 'AtTolerance')

Cloning the Group

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

6.1.4.3.5.2 Limit<Limit>

RepCap Settings

```
# Range: Nr1 .. Nr12
rc = driver.configure.multiEval.limit.seMask.limit.repcap_limit_get()
driver.configure.multiEval.limit.seMask.limit.repcap_limit_set(repcap.Limit.Nr1)
```

class LimitCls

Limit commands group definition. 8 total commands, 3 Subgroups, 0 group commands Repeated Capability: Limit, default value after init: Limit.Nr1

Cloning the Group

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

Subgroups

6.1.4.3.5.3 Additional<Table>

RepCap Settings

```
# Range: Nr1 .. Nr5
rc = driver.configure.multiEval.limit.seMask.limit.additional.repcap_table_get()
driver.configure.multiEval.limit.seMask.limit.additional.repcap_table_set(repcap.Table.
↪Nr1)
```

class AdditionalCls

Additional commands group definition. 4 total commands, 2 Subgroups, 0 group commands Repeated Capability: Table, default value after init: Table.Nr1

Cloning the Group

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

Subgroups

6.1.4.3.5.4 CarrierAggregation

class CarrierAggregationCls

CarrierAggregation 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.configure.multiEval.limit.seMask.limit.additional.carrierAggregation.
↪clone()
```

Subgroups

6.1.4.3.5.5 ChannelBw1st<FirstChannelBw>

RepCap Settings

```
# Range: Bw100 .. Bw200
rc = driver.configure.multiEval.limit.seMask.limit.additional.carrierAggregation.
↪channelBw1st.repcap_firstChannelBw_get()
driver.configure.multiEval.limit.seMask.limit.additional.carrierAggregation.channelBw1st.
↪repcap_firstChannelBw_set(repcap.FirstChannelBw.Bw100)
```

class ChannelBw1stCls

ChannelBw1st commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability: FirstChannelBw, default value after init: FirstChannelBw.Bw100

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.seMask.limit.additional.carrierAggregation.
↪channelBw1st.clone()
```

Subgroups

6.1.4.3.5.6 ChannelBw2nd<SecondChannelBw>

RepCap Settings

```
# Range: Bw50 .. Bw200
rc = driver.configure.multiEval.limit.seMask.limit.additional.carrierAggregation.
↪channelBw1st.channelBw2nd.repcap_secondChannelBw_get()
driver.configure.multiEval.limit.seMask.limit.additional.carrierAggregation.channelBw1st.
↪channelBw2nd.repcap_secondChannelBw_set(repcap.SecondChannelBw.Bw50)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>:ADDitional<Table>
↪:CAGGregation:CBANdwidth<Band1>:CBANdwidth<Band2>
```

class ChannelBw2ndCls

ChannelBw2nd commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: SecondChannelBw, default value after init: SecondChannelBw.Bw50

class ChannelBw2ndStruct

Response structure. Fields:

- Enable: bool: OFF | ON OFF: Disables the check of these requirements. ON: Enables the check of these requirements.
- Frequency_Start: float: numeric Start frequency of the area, relative to the edges of the aggregated channel bandwidth. Range: 0 MHz to 65 MHz, Unit: Hz
- Frequency_End: float: numeric Stop frequency of the area, relative to the edges of the aggregated channel bandwidth. Range: 0 MHz to 65 MHz, Unit: Hz
- Level: float: numeric Upper limit for the area Range: -256 dBm to 256 dBm, Unit: dBm
- Rbw: enums.Rbw: K030 | K100 | M1 Resolution bandwidth to be used for the area. K030: 30 kHz K100: 100 kHz M1: 1 MHz

get(*limit=Limit.Default*, *table=Table.Default*, *firstChannelBw=FirstChannelBw.Default*, *secondChannelBw=SecondChannelBw.Default*) → ChannelBw2ndStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>
↪:ADDitional<Table>:CAGGregation:CBANdwidth<Band1>:CBANdwidth<Band2>
value: ChannelBw2ndStruct = driver.configure.multiEval.limit.seMask.limit.
↪additional.carrierAggregation.channelBw1st.channelBw2nd.get(limit = repcap.
↪Limit.Default, table = repcap.Table.Default, firstChannelBw = repcap.
↪FirstChannelBw.Default, secondChannelBw = repcap.SecondChannelBw.Default)
```

Defines additional requirements for the emission mask area <no>. The activation state, the area borders, an upper limit and the resolution bandwidth must be specified. The settings are defined separately for each channel bandwidth combination, for two aggregated carriers. The following bandwidth combinations are supported: Example: For the first line in the figure, use …:CBANdwidth100:CBANdwidth50.

param limit

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param table
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Additional')

param firstChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw1st')

param secondChannelBw
optional repeated capability selector. Default value: Bw50 (settable in the interface 'ChannelBw2nd')

return
structure: for return value, see the help for ChannelBw2ndStruct structure arguments.

set(*enable: bool*, *frequency_start: float*, *frequency_end: float*, *level: float*, *rbw: Rbw*, *limit=Limit.Default*, *table=Table.Default*, *firstChannelBw=FirstChannelBw.Default*, *secondChannelBw=SecondChannelBw.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>
↪:ADDitional<Table>:CAGGregation:CBANdwidth<Band1>:CBANdwidth<Band2>
driver.configure.multiEval.limit.seMask.limit.additional.carrierAggregation.
↪channelBw1st.channelBw2nd.set(enable = False, frequency_start = 1.0,␣
↪frequency_end = 1.0, level = 1.0, rbw = enums.Rbw.K030, limit = repcap.Limit.
↪Default, table = repcap.Table.Default, firstChannelBw = repcap.FirstChannelBw.
↪Default, secondChannelBw = repcap.SecondChannelBw.Default)
```

Defines additional requirements for the emission mask area <no>. The activation state, the area borders, an upper limit and the resolution bandwidth must be specified. The settings are defined separately for each channel bandwidth combination, for two aggregated carriers. The following bandwidth combinations are supported: Example: For the first line in the figure, use ...:CBANdwidth100:CBANdwidth50.

param enable
OFF | ON OFF: Disables the check of these requirements. ON: Enables the check of these requirements.

param frequency_start
numeric Start frequency of the area, relative to the edges of the aggregated channel bandwidth. Range: 0 MHz to 65 MHz, Unit: Hz

param frequency_end
numeric Stop frequency of the area, relative to the edges of the aggregated channel bandwidth. Range: 0 MHz to 65 MHz, Unit: Hz

param level
numeric Upper limit for the area Range: -256 dBm to 256 dBm, Unit: dBm

param rbw
K030 | K100 | M1 Resolution bandwidth to be used for the area. K030: 30 kHz K100: 100 kHz M1: 1 MHz

param limit
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param table
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Additional')

param firstChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw1st')

param secondChannelBw
optional repeated capability selector. Default value: Bw50 (settable in the interface 'ChannelBw2nd')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.seMask.limit.additional.carrierAggregation.
↪channelBw1st.channelBw2nd.clone()
```

6.1.4.3.5.7 Ocombination

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>:ADDitional<Table>
↪:CAGGregation:OCOMbination
```

class OcombinationCls

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

class OcombinationStruct

Response structure. Fields:

- Enable: bool: OFF | ON OFF: Disables the check of these requirements. ON: Enables the check of these requirements.
- Frequency_Start: float: numeric Start frequency of the area, relative to the edges of the aggregated channel bandwidth. Range: 0 MHz to 65 MHz, Unit: Hz
- Frequency_End: float: numeric Stop frequency of the area, relative to the edges of the aggregated channel bandwidth. Range: 0 MHz to 65 MHz, Unit: Hz
- Level: float: numeric Upper limit for the area. Range: -256 dBm to 256 dBm, Unit: dBm
- Rbw: enums.Rbw: K030 | K100 | M1 Resolution bandwidth to be used for the area. K030: 30 kHz K100: 100 kHz M1: 1 MHz

get(*limit=Limit.Default*, *table=Table.Default*) → OcombinationStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>
↪:ADDitional<Table>:CAGGregation:OCOMbination
value: OcombinationStruct = driver.configure.multiEval.limit.seMask.limit.
↪additional.carrierAggregation.ocombination.get(limit = repcap.Limit.Default,␣
↪table = repcap.Table.Default)
```

Defines additional requirements for the emission mask area <no>. The activation state, the area borders, an upper limit and the resolution bandwidth must be specified. The settings apply to all 'other' channel bandwidth combinations, not covered by other commands in this chapter.

param limit
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param table
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Additional')

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

set(*enable: bool*, *frequency_start: float*, *frequency_end: float*, *level: float*, *rbw: Rbw*, *limit=Limit.Default*, *table=Table.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>
↪:ADDitional<Table>:CAGGregation:OCOMbination
driver.configure.multiEval.limit.seMask.limit.additional.carrierAggregation.
↪ocombination.set(enable = False, frequency_start = 1.0, frequency_end = 1.0,␣
↪level = 1.0, rbw = enums.Rbw.K030, limit = repcap.Limit.Default, table =␣
↪repcap.Table.Default)
```

Defines additional requirements for the emission mask area <no>. The activation state, the area borders, an upper limit and the resolution bandwidth must be specified. The settings apply to all 'other' channel bandwidth combinations, not covered by other commands in this chapter.

param enable
OFF | ON OFF: Disables the check of these requirements. ON: Enables the check of these requirements.

param frequency_start
numeric Start frequency of the area, relative to the edges of the aggregated channel bandwidth. Range: 0 MHz to 65 MHz, Unit: Hz

param frequency_end
numeric Stop frequency of the area, relative to the edges of the aggregated channel bandwidth. Range: 0 MHz to 65 MHz, Unit: Hz

param level
numeric Upper limit for the area. Range: -256 dBm to 256 dBm, Unit: dBm

param rbw
K030 | K100 | M1 Resolution bandwidth to be used for the area. K030: 30 kHz K100: 100 kHz M1: 1 MHz

param limit
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param table
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Additional')

6.1.4.3.5.8 ChannelBw<ChannelBw>

RepCap Settings

```
# Range: Bw14 .. Bw200
rc = driver.configure.multiEval.limit.seMask.limit.additional.channelBw.repcap_channelBw_
↪get()
driver.configure.multiEval.limit.seMask.limit.additional.channelBw.repcap_channelBw_
↪set(repcap.ChannelBw.Bw14)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>:ADDitional<Table>
↪:CBANdwidth<Band>
```

class ChannelBwCls

ChannelBw commands group definition. 2 total commands, 1 Subgroups, 1 group commands Repeated Capability: ChannelBw, default value after init: ChannelBw.Bw14

class ChannelBwStruct

Response structure. Fields:

- Enable: bool: OFF | ON OFF: Disables the check of these requirements. ON: Enables the check of these requirements.
- Frequency_Start: float: numeric Start frequency of the area, relative to the edges of the channel bandwidth. Range: see table below , Unit: Hz
- Frequency_End: float: numeric Stop frequency of the area, relative to the edges of the channel bandwidth. Range: see table below , Unit: Hz
- Level: float: numeric Upper limit for the area Range: -256 dBm to 256 dBm, Unit: dBm
- Rbw: enums.RbwExtended: K030 | K050 | K100 | K150 | K200 | M1 Resolution bandwidth to be used for the area. Only a subset of the values is allowed, depending on Table and Band, see table below. K030: 30 kHz K050: 50 kHz K100: 100 kHz K150: 150 kHz K200: 200 kHz M1: 1 MHz

get(*limit=Limit.Default*, *table=Table.Default*, *channelBw=ChannelBw.Default*) → ChannelBwStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>
↪:ADDitional<Table>:CBANdwidth<Band>
value: ChannelBwStruct = driver.configure.multiEval.limit.seMask.limit.
↪additional.channelBw.get(limit = repcap.Limit.Default, table = repcap.Table.
↪Default, channelBw = repcap.ChannelBw.Default)
```

Defines additional requirements for the emission mask area <no>, for uplink measurements. The activation state, the area borders, an upper limit and the resolution bandwidth must be specified. The emission mask applies to the channel bandwidth <Band>. Several tables of additional requirements are available.

param limit
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param table
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Additional')

param channelBw
optional repeated capability selector. Default value: Bw14 (settable in the interface 'ChannelBw')

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

set(*enable: bool*, *frequency_start: float*, *frequency_end: float*, *level: float*, *rbw: RbwExtended*, *limit=Limit.Default*, *table=Table.Default*, *channelBw=ChannelBw.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>
↪:ADDitional<Table>:CBANdwidth<Band>
driver.configure.multiEval.limit.seMask.limit.additional.channelBw.set(enable =␣
↪False, frequency_start = 1.0, frequency_end = 1.0, level = 1.0, rbw = enums.
↪RbwExtended.K030, limit = repcap.Limit.Default, table = repcap.Table.Default,␣
↪channelBw = repcap.ChannelBw.Default)
```

Defines additional requirements for the emission mask area <no>, for uplink measurements. The activation state, the area borders, an upper limit and the resolution bandwidth must be specified. The emission mask applies to the channel bandwidth <Band>. Several tables of additional requirements are available.

param enable
OFF | ON OFF: Disables the check of these requirements. ON: Enables the check of these requirements.

param frequency_start
numeric Start frequency of the area, relative to the edges of the channel bandwidth. Range: see table below , Unit: Hz

param frequency_end
numeric Stop frequency of the area, relative to the edges of the channel bandwidth. Range: see table below , Unit: Hz

param level
numeric Upper limit for the area Range: -256 dBm to 256 dBm, Unit: dBm

param rbw
K030 | K050 | K100 | K150 | K200 | M1 Resolution bandwidth to be used for the area. Only a subset of the values is allowed, depending on Table and Band, see table below. K030: 30 kHz K050: 50 kHz K100: 100 kHz K150: 150 kHz K200: 200 kHz M1: 1 MHz

param limit
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param table
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Additional')

param channelBw
optional repeated capability selector. Default value: Bw14 (settable in the interface 'ChannelBw')

Cloning the Group

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

Subgroups

6.1.4.3.5.9 Sidelink

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>:ADDitional<Table>
↪:CBANdwidth<Band>:SIDelink
```

class SidelinkCls

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

class SidelinkStruct

Structure for setting input parameters. Fields:

- Enable: bool: OFF | ON OFF: Disables the check of these requirements. ON: Enables the check of these requirements.
- Frequency_Start: float: numeric Start frequency of the area, relative to the edges of the channel bandwidth. Range: 0 MHz to 25 MHz, Unit: Hz
- Frequency_End: float: numeric Stop frequency of the area, relative to the edges of the channel bandwidth. Range: 0 MHz to 25 MHz, Unit: Hz
- Level: float: numeric Upper limit at FrequencyStart Range: -256 dBm to 256 dBm, Unit: dBm
- Slope: float: numeric Slope for the upper limit within the area Range: -256 dB/MHz to 256 dB/MHz, Unit: dB/MHz
- Rbw: enums.Rbw: K030 | K100 | M1 Resolution bandwidth to be used for the area. K030: 30 kHz K100: 100 kHz M1: 1 MHz

get(*limit=Limit.Default*, *table=Table.Default*, *channelBw=ChannelBw.Default*) → SidelinkStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>
↪:ADDitional<Table>:CBANdwidth<Band>:SIDelink
value: SidelinkStruct = driver.configure.multiEval.limit.seMask.limit.
↪additional.channelBw.sidelink.get(limit = repcap.Limit.Default, table =␣
↪repcap.Table.Default, channelBw = repcap.ChannelBw.Default)
```

Defines additional requirements for the emission mask area <no>, for sidelink measurements. The activation state, the area borders, the start value and slope of the upper limit and the resolution bandwidth must be specified. The emission mask applies to the channel bandwidth <Band>.

param limit
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param table
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Additional')

param channelBw
optional repeated capability selector. Default value: Bw14 (settable in the interface 'ChannelBw')

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

set(*structure: SidelinkStruct*, *limit=Limit.Default*, *table=Table.Default*, *channelBw=ChannelBw.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>
↪:ADDitional<Table>:CBANdwidth<Band>:SIDelink
structure = driver.configure.multiEval.limit.seMask.limit.additional.channelBw.
↪sidelink.SidelinkStruct()
structure.Enable: bool = False
structure.Frequency_Start: float = 1.0
structure.Frequency_End: float = 1.0
structure.Level: float = 1.0
structure.Slope: float = 1.0
structure.Rbw: enums.Rbw = enums.Rbw.K030
driver.configure.multiEval.limit.seMask.limit.additional.channelBw.sidelink.
↪set(structure, limit = repcap.Limit.Default, table = repcap.Table.Default,␣
↪channelBw = repcap.ChannelBw.Default)
```

Defines additional requirements for the emission mask area <no>, for sidelink measurements. The activation state, the area borders, the start value and slope of the upper limit and the resolution bandwidth must be specified. The emission mask applies to the channel bandwidth <Band>.

param structure
for set value, see the help for SidelinkStruct structure arguments.

param limit
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param table
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Additional')

param channelBw
optional repeated capability selector. Default value: Bw14 (settable in the interface 'ChannelBw')

6.1.4.3.5.10 CarrierAggregation

class CarrierAggregationCls

CarrierAggregation commands group definition. 3 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.1.4.3.5.11 ChannelBw1st<FirstChannelBw>

RepCap Settings

```
# Range: Bw100 .. Bw200
rc = driver.configure.multiEval.limit.seMask.limit.carrierAggregation.channelBw1st.
↪repcap_firstChannelBw_get()
driver.configure.multiEval.limit.seMask.limit.carrierAggregation.channelBw1st.repcap_
↪firstChannelBw_set(repcap.FirstChannelBw.Bw100)
```

class ChannelBw1stCls

ChannelBw1st commands group definition. 2 total commands, 1 Subgroups, 0 group commands Repeated Capability: FirstChannelBw, default value after init: FirstChannelBw.Bw100

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.seMask.limit.carrierAggregation.channelBw1st.
↪clone()
```

Subgroups

6.1.4.3.5.12 ChannelBw2nd<SecondChannelBw>

RepCap Settings

```
# Range: Bw50 .. Bw200
rc = driver.configure.multiEval.limit.seMask.limit.carrierAggregation.channelBw1st.
↪channelBw2nd.repcap_secondChannelBw_get()
driver.configure.multiEval.limit.seMask.limit.carrierAggregation.channelBw1st.
↪channelBw2nd.repcap_secondChannelBw_set(repcap.SecondChannelBw.Bw50)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>
↪:CAGGregation:CBANdwidth<Band1>:CBANdwidth<Band2>
```

class ChannelBw2ndCls

ChannelBw2nd commands group definition. 2 total commands, 1 Subgroups, 1 group commands Repeated Capability: SecondChannelBw, default value after init: SecondChannelBw.Bw50

class ChannelBw2ndStruct

Response structure. Fields:

- Enable: bool: OFF | ON OFF: Disables the check of these requirements. ON: Enables the check of these requirements.
- Frequency_Start: float: numeric Start frequency of the area, relative to the edges of the aggregated channel bandwidth. Range: 0 MHz to 65 MHz, Unit: Hz
- Frequency_End: float: numeric Stop frequency of the area, relative to the edges of the aggregated channel bandwidth. Range: 0 MHz to 65 MHz, Unit: Hz
- Level: float: numeric Upper limit for the area Range: -256 dBm to 256 dBm, Unit: dBm
- Rbw: enums.Rbw: K030 | K100 | M1 Resolution bandwidth to be used for the area. K030: 30 kHz K100: 100 kHz M1: 1 MHz

get(*limit=Limit.Default*, *firstChannelBw=FirstChannelBw.Default*, *secondChannelBw=SecondChannelBw.Default*) → ChannelBw2ndStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>
↪:CAGGregation:CBANdwidth<Band1>:CBANdwidth<Band2>
value: ChannelBw2ndStruct = driver.configure.multiEval.limit.seMask.limit.
↪carrierAggregation.channelBw1st.channelBw2nd.get(limit = repcap.Limit.Default,
↪ firstChannelBw = repcap.FirstChannelBw.Default, secondChannelBw = repcap.
↪SecondChannelBw.Default)
```

Defines general requirements for the emission mask area <no>. The activation state, the area borders, an upper limit and the resolution bandwidth must be specified. The settings are defined separately for each channel bandwidth combination, for two aggregated carriers. The following bandwidth combinations are supported: Example: For the first line in the figure, use …:CBANdwidth100:CBANdwidth50.

param limit
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param firstChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw1st')

param secondChannelBw
optional repeated capability selector. Default value: Bw50 (settable in the interface 'ChannelBw2nd')

return
structure: for return value, see the help for ChannelBw2ndStruct structure arguments.

set(*enable: bool*, *frequency_start: float*, *frequency_end: float*, *level: float*, *rbw: Rbw*, *limit=Limit.Default*, *firstChannelBw=FirstChannelBw.Default*, *secondChannelBw=SecondChannelBw.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>
↪:CAGGregation:CBANdwidth<Band1>:CBANdwidth<Band2>
driver.configure.multiEval.limit.seMask.limit.carrierAggregation.channelBw1st.
↪channelBw2nd.set(enable = False, frequency_start = 1.0, frequency_end = 1.0,␣
↪level = 1.0, rbw = enums.Rbw.K030, limit = repcap.Limit.Default,␣
↪firstChannelBw = repcap.FirstChannelBw.Default, secondChannelBw = repcap.
↪SecondChannelBw.Default)
```

Defines general requirements for the emission mask area <no>. The activation state, the area borders, an upper limit and the resolution bandwidth must be specified. The settings are defined separately for each channel bandwidth combination, for two aggregated carriers. The following bandwidth combinations are supported: Example: For the first line in the figure, use ...:CBANdwidth100:CBANdwidth50.

param enable
OFF | ON OFF: Disables the check of these requirements. ON: Enables the check of these requirements.

param frequency_start
numeric Start frequency of the area, relative to the edges of the aggregated channel bandwidth. Range: 0 MHz to 65 MHz, Unit: Hz

param frequency_end
numeric Stop frequency of the area, relative to the edges of the aggregated channel bandwidth. Range: 0 MHz to 65 MHz, Unit: Hz

param level
numeric Upper limit for the area Range: -256 dBm to 256 dBm, Unit: dBm

param rbw
K030 | K100 | M1 Resolution bandwidth to be used for the area. K030: 30 kHz K100: 100 kHz M1: 1 MHz

param limit
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param firstChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw1st')

param secondChannelBw
optional repeated capability selector. Default value: Bw50 (settable in the interface 'ChannelBw2nd')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.seMask.limit.carrierAggregation.channelBw1st.
↪channelBw2nd.clone()
```

Subgroups

6.1.4.3.5.13 ChannelBw3rd<ThirdChannelBw>

RepCap Settings

```
# Range: Bw100 .. Bw200
rc = driver.configure.multiEval.limit.seMask.limit.carrierAggregation.channelBw1st.
↪channelBw2nd.channelBw3rd.repcap_thirdChannelBw_get()
driver.configure.multiEval.limit.seMask.limit.carrierAggregation.channelBw1st.
↪channelBw2nd.channelBw3rd.repcap_thirdChannelBw_set(repcap.ThirdChannelBw.Bw100)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>
↪:CAGGregation:CBANdwidth<Band1>:CBANdwidth<Band2>:CBANdwidth<Band3>
```

class ChannelBw3rdCls

ChannelBw3rd commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: ThirdChannelBw, default value after init: ThirdChannelBw.Bw100

class ChannelBw3rdStruct

Response structure. Fields:

- Enable: bool: OFF | ON OFF: Disables the check of these requirements. ON: Enables the check of these requirements.
- Frequency_Start: float: numeric Start frequency of the area, relative to the edges of the aggregated channel bandwidth. Range: 0 MHz to 65 MHz, Unit: Hz
- Frequency_End: float: numeric Stop frequency of the area, relative to the edges of the aggregated channel bandwidth. Range: 0 MHz to 65 MHz, Unit: Hz
- Level: float: numeric Upper limit for the area Range: -256 dBm to 256 dBm, Unit: dBm
- Rbw: enums.Rbw: K030 | K100 | M1 Resolution bandwidth to be used for the area. K030: 30 kHz K100: 100 kHz M1: 1 MHz

get(*limit=Limit.Default*, *firstChannelBw=FirstChannelBw.Default*, *secondChannelBw=SecondChannelBw.Default*, *thirdChannelBw=ThirdChannelBw.Default*) → ChannelBw3rdStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>
↪:CAGGregation:CBANdwidth<Band1>:CBANdwidth<Band2>:CBANdwidth<Band3>
value: ChannelBw3rdStruct = driver.configure.multiEval.limit.seMask.limit.
↪carrierAggregation.channelBw1st.channelBw2nd.channelBw3rd.get(limit = repcap.
↪Limit.Default, firstChannelBw = repcap.FirstChannelBw.Default,␣
↪secondChannelBw = repcap.SecondChannelBw.Default, thirdChannelBw = repcap.
↪ThirdChannelBw.Default)
```

Defines general requirements for the emission mask area <no>. The activation state, the area borders, an upper limit and the resolution bandwidth must be specified. The settings are defined separately for each channel bandwidth combination, for three aggregated carriers. The following bandwidth combinations are supported: Example: For the first line in the figure, use …:CBANdwidth200:CBANdwidth150:CBANdwidth100.

param limit
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param firstChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw1st')

param secondChannelBw
optional repeated capability selector. Default value: Bw50 (settable in the interface 'ChannelBw2nd')

param thirdChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw3rd')

return
structure: for return value, see the help for ChannelBw3rdStruct structure arguments.

set(*enable: bool*, *frequency_start: float*, *frequency_end: float*, *level: float*, *rbw: Rbw*, *limit=Limit.Default*, *firstChannelBw=FirstChannelBw.Default*, *secondChannelBw=SecondChannelBw.Default*, *thirdChannelBw=ThirdChannelBw.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>
↪:CAGGregation:CBANdwidth<Band1>:CBANdwidth<Band2>:CBANdwidth<Band3>
driver.configure.multiEval.limit.seMask.limit.carrierAggregation.channelBw1st.
↪channelBw2nd.channelBw3rd.set(enable = False, frequency_start = 1.0,␣
↪frequency_end = 1.0, level = 1.0, rbw = enums.Rbw.K030, limit = repcap.Limit.
↪Default, firstChannelBw = repcap.FirstChannelBw.Default, secondChannelBw =␣
↪repcap.SecondChannelBw.Default, thirdChannelBw = repcap.ThirdChannelBw.
↪Default)
```

Defines general requirements for the emission mask area <no>. The activation state, the area borders, an upper limit and the resolution bandwidth must be specified. The settings are defined separately for each channel bandwidth combination, for three aggregated carriers. The following bandwidth combinations are supported: Example: For the first line in the figure, use ...:CBANdwidth200:CBANdwidth150:CBANdwidth100.

param enable
OFF | ON OFF: Disables the check of these requirements. ON: Enables the check of these requirements.

param frequency_start
numeric Start frequency of the area, relative to the edges of the aggregated channel bandwidth. Range: 0 MHz to 65 MHz, Unit: Hz

param frequency_end
numeric Stop frequency of the area, relative to the edges of the aggregated channel bandwidth. Range: 0 MHz to 65 MHz, Unit: Hz

param level
numeric Upper limit for the area Range: -256 dBm to 256 dBm, Unit: dBm

param rbw
K030 | K100 | M1 Resolution bandwidth to be used for the area. K030: 30 kHz K100: 100 kHz M1: 1 MHz

param limit
optional repeated capability selector. Default value: Nr1 (settable in the interface

'Limit')

param firstChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw1st')

param secondChannelBw
optional repeated capability selector. Default value: Bw50 (settable in the interface 'ChannelBw2nd')

param thirdChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw3rd')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.seMask.limit.carrierAggregation.channelBw1st.
↪channelBw2nd.channelBw3rd.clone()
```

6.1.4.3.5.14 Ocombination

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>
↪:CAGGregation:OCOMbination
```

class OcombinationCls

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

class OcombinationStruct

Response structure. Fields:

- Enable: bool: OFF | ON OFF: Disables the check of these requirements. ON: Enables the check of these requirements.
- Frequency_Start: float: numeric Start frequency of the area, relative to the edges of the aggregated channel bandwidth. Range: 0 MHz to 65 MHz, Unit: Hz
- Frequency_End: float: numeric Stop frequency of the area, relative to the edges of the aggregated channel bandwidth. Range: 0 MHz to 65 MHz, Unit: Hz
- Level: float: numeric Upper limit for the area Range: -256 dBm to 256 dBm, Unit: dBm
- Rbw: enums.Rbw: K030 | K100 | M1 Resolution bandwidth to be used for the area. K030: 30 kHz K100: 100 kHz M1: 1 MHz

get(*limit=Limit.Default*) → OcombinationStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>
↪:CAGGregation:OCOMbination
value: OcombinationStruct = driver.configure.multiEval.limit.seMask.limit.
↪carrierAggregation.ocombination.get(limit = repcap.Limit.Default)
```

Defines general requirements for the emission mask area <no>. The activation state, the area borders, an upper limit and the resolution bandwidth must be specified. The settings apply to all 'other' channel bandwidth combinations, not covered by other commands in this chapter.

param limit
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

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

set(*enable: bool*, *frequency_start: float*, *frequency_end: float*, *level: float*, *rbw: Rbw*, *limit=Limit.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>
↪:CAGGregation:OCOMbination
driver.configure.multiEval.limit.seMask.limit.carrierAggregation.ocombination.
↪set(enable = False, frequency_start = 1.0, frequency_end = 1.0, level = 1.0,␣
↪rbw = enums.Rbw.K030, limit = repcap.Limit.Default)
```

Defines general requirements for the emission mask area <no>. The activation state, the area borders, an upper limit and the resolution bandwidth must be specified. The settings apply to all 'other' channel bandwidth combinations, not covered by other commands in this chapter.

param enable
OFF | ON OFF: Disables the check of these requirements. ON: Enables the check of these requirements.

param frequency_start
numeric Start frequency of the area, relative to the edges of the aggregated channel bandwidth. Range: 0 MHz to 65 MHz, Unit: Hz

param frequency_end
numeric Stop frequency of the area, relative to the edges of the aggregated channel bandwidth. Range: 0 MHz to 65 MHz, Unit: Hz

param level
numeric Upper limit for the area Range: -256 dBm to 256 dBm, Unit: dBm

param rbw
K030 | K100 | M1 Resolution bandwidth to be used for the area. K030: 30 kHz K100: 100 kHz M1: 1 MHz

param limit
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

6.1.4.3.5.15 ChannelBw<ChannelBw>

RepCap Settings

```
# Range: Bw14 .. Bw200
rc = driver.configure.multiEval.limit.seMask.limit.channelBw.repcap_channelBw_get()
driver.configure.multiEval.limit.seMask.limit.channelBw.repcap_channelBw_set(repcap.
↪ChannelBw.Bw14)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>:CBANdwidth<Band>
```

class ChannelBwCls

ChannelBw commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: ChannelBw, default value after init: ChannelBw.Bw14

class ChannelBwStruct

Response structure. Fields:

- Enable: bool: OFF | ON OFF: Disables the check of these requirements. ON: Enables the check of these requirements.
- Frequency_Start: float: numeric Start frequency of the area, relative to the edges of the channel bandwidth. Range: see table below , Unit: Hz
- Frequency_End: float: numeric Stop frequency of the area, relative to the edges of the channel bandwidth. Range: see table below , Unit: Hz
- Level: float: numeric Upper limit for the area Range: -256 dBm to 256 dBm, Unit: dBm
- Rbw: enums.Rbw: K030 | K100 | M1 Resolution bandwidth to be used for the area. K030: 30 kHz K100: 100 kHz M1: 1 MHz

get(*limit=Limit.Default*, *channelBw=ChannelBw.Default*) → ChannelBwStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>
↪:CBANdwidth<Band>
value: ChannelBwStruct = driver.configure.multiEval.limit.seMask.limit.
↪channelBw.get(limit = repcap.Limit.Default, channelBw = repcap.ChannelBw.
↪Default)
```

Defines general requirements for the emission mask area <no>. The activation state, the area borders, an upper limit and the resolution bandwidth must be specified. The emission mask applies to the channel bandwidth <Band>.

param limit
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param channelBw
optional repeated capability selector. Default value: Bw14 (settable in the interface 'ChannelBw')

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

set(*enable: bool*, *frequency_start: float*, *frequency_end: float*, *level: float*, *rbw: Rbw*, *limit=Limit.Default*, *channelBw=ChannelBw.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>
↪:CBANdwidth<Band>
driver.configure.multiEval.limit.seMask.limit.channelBw.set(enable = False,␣
↪frequency_start = 1.0, frequency_end = 1.0, level = 1.0, rbw = enums.Rbw.K030,
↪ limit = repcap.Limit.Default, channelBw = repcap.ChannelBw.Default)
```

Defines general requirements for the emission mask area <no>. The activation state, the area borders, an upper limit and the resolution bandwidth must be specified. The emission mask applies to the channel bandwidth <Band>.

param enable
OFF | ON OFF: Disables the check of these requirements. ON: Enables the check of these requirements.

param frequency_start
numeric Start frequency of the area, relative to the edges of the channel bandwidth. Range: see table below , Unit: Hz

param frequency_end
numeric Stop frequency of the area, relative to the edges of the channel bandwidth. Range: see table below , Unit: Hz

param level
numeric Upper limit for the area Range: -256 dBm to 256 dBm, Unit: dBm

param rbw
K030 | K100 | M1 Resolution bandwidth to be used for the area. K030: 30 kHz K100: 100 kHz M1: 1 MHz

param limit
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param channelBw
optional repeated capability selector. Default value: Bw14 (settable in the interface 'ChannelBw')

Cloning the Group

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

6.1.4.3.5.16 ObwLimit

class ObwLimitCls

ObwLimit 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.configure.multiEval.limit.seMask.obwLimit.clone()
```

Subgroups

6.1.4.3.5.17 CarrierAggregation

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:OBWLimit:CAGGregation:OCOMbination
```

class CarrierAggregationCls

CarrierAggregation commands group definition. 3 total commands, 1 Subgroups, 1 group commands

get_ocombination() → float

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:OBWLimit:CAGGregation:OCOMbination
value: float or bool = driver.configure.multiEval.limit.seMask.obwLimit.
↪carrierAggregation.get_ocombination()
```

Defines an upper limit for the occupied bandwidth. The setting applies to all 'other' channel bandwidth combinations, not covered by other commands in this chapter.

return
obw_limit: (float or boolean) numeric | ON | OFF Range: 0 MHz to 40 MHz, Unit: Hz ON | OFF enables or disables the limit check.

set_ocombination(*obw_limit: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:OBWLimit:CAGGregation:OCOMbination
driver.configure.multiEval.limit.seMask.obwLimit.carrierAggregation.set_
↪ocombination(obw_limit = 1.0)
```

Defines an upper limit for the occupied bandwidth. The setting applies to all 'other' channel bandwidth combinations, not covered by other commands in this chapter.

param obw_limit
(float or boolean) numeric | ON | OFF Range: 0 MHz to 40 MHz, Unit: Hz ON | OFF enables or disables the limit check.

Cloning the Group

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

Subgroups

6.1.4.3.5.18 ChannelBw1st<FirstChannelBw>

RepCap Settings

```
# Range: Bw100 .. Bw200
rc = driver.configure.multiEval.limit.seMask.obwLimit.carrierAggregation.channelBw1st.
↪repcap_firstChannelBw_get()
driver.configure.multiEval.limit.seMask.obwLimit.carrierAggregation.channelBw1st.repcap_
↪firstChannelBw_set(repcap.FirstChannelBw.Bw100)
```

class ChannelBw1stCls

ChannelBw1st commands group definition. 2 total commands, 1 Subgroups, 0 group commands Repeated Capability: FirstChannelBw, default value after init: FirstChannelBw.Bw100

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.seMask.obwLimit.carrierAggregation.
↪channelBw1st.clone()
```

Subgroups

6.1.4.3.5.19 ChannelBw2nd<SecondChannelBw>

RepCap Settings

```
# Range: Bw50 .. Bw200
rc = driver.configure.multiEval.limit.seMask.obwLimit.carrierAggregation.channelBw1st.
↪channelBw2nd.repcap_secondChannelBw_get()
driver.configure.multiEval.limit.seMask.obwLimit.carrierAggregation.channelBw1st.
↪channelBw2nd.repcap_secondChannelBw_set(repcap.SecondChannelBw.Bw50)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:OBWLimit:CAGGregation:CBANdwidth<Band1>:CBANdwidth<Band2>
```

class ChannelBw2ndCls

ChannelBw2nd commands group definition. 2 total commands, 1 Subgroups, 1 group commands Repeated Capability: SecondChannelBw, default value after init: SecondChannelBw.Bw50

get(*firstChannelBw=FirstChannelBw.Default*, *secondChannelBw=SecondChannelBw.Default*) → float

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:OBWLimit:CAGGregation:CBANdwidth<Band1>:CBANdwidth
↪<Band2>
```

(continues on next page)

(continued from previous page)

```
value: float or bool = driver.configure.multiEval.limit.seMask.obwLimit.
↪carrierAggregation.channelBw1st.channelBw2nd.get(firstChannelBw = repcap.
↪FirstChannelBw.Default, secondChannelBw = repcap.SecondChannelBw.Default)
```

Defines an upper limit for the occupied bandwidth. The setting is defined separately for each channel bandwidth combination, for two aggregated carriers. The following bandwidth combinations are supported: Example: For the first line in the figure, use ...:CBANdwidth100:CBANdwidth50.

param firstChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw1st')

param secondChannelBw
optional repeated capability selector. Default value: Bw50 (settable in the interface 'ChannelBw2nd')

return
obw_limit: (float or boolean) numeric | ON | OFF Range: 0 MHz to 40 MHz, Unit: Hz ON | OFF enables or disables the limit check.

set(*obw_limit: float*, *firstChannelBw=FirstChannelBw.Default*, *secondChannelBw=SecondChannelBw.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:OBWLimit:CAGGregation:CBANdwidth<Band1>:CBANdwidth
↪<Band2>
driver.configure.multiEval.limit.seMask.obwLimit.carrierAggregation.
↪channelBw1st.channelBw2nd.set(obw_limit = 1.0, firstChannelBw = repcap.
↪FirstChannelBw.Default, secondChannelBw = repcap.SecondChannelBw.Default)
```

Defines an upper limit for the occupied bandwidth. The setting is defined separately for each channel bandwidth combination, for two aggregated carriers. The following bandwidth combinations are supported: Example: For the first line in the figure, use ...:CBANdwidth100:CBANdwidth50.

param obw_limit
(float or boolean) numeric | ON | OFF Range: 0 MHz to 40 MHz, Unit: Hz ON | OFF enables or disables the limit check.

param firstChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw1st')

param secondChannelBw
optional repeated capability selector. Default value: Bw50 (settable in the interface 'ChannelBw2nd')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.seMask.obwLimit.carrierAggregation.
↪channelBw1st.channelBw2nd.clone()
```

Subgroups

6.1.4.3.5.20 ChannelBw3rd<ThirdChannelBw>

RepCap Settings

```
# Range: Bw100 .. Bw200
rc = driver.configure.multiEval.limit.seMask.obwLimit.carrierAggregation.channelBw1st.
↪channelBw2nd.channelBw3rd.repcap_thirdChannelBw_get()
driver.configure.multiEval.limit.seMask.obwLimit.carrierAggregation.channelBw1st.
↪channelBw2nd.channelBw3rd.repcap_thirdChannelBw_set(repcap.ThirdChannelBw.Bw100)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:OBWLimit:CAGGregation:CBANdwidth<Band1>:CBANdwidth<Band2>
↪:CBANdwidth<Band3>
```

class ChannelBw3rdCls

ChannelBw3rd commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: ThirdChannelBw, default value after init: ThirdChannelBw.Bw100

get(*firstChannelBw=FirstChannelBw.Default*, *secondChannelBw=SecondChannelBw.Default*, *thirdChannelBw=ThirdChannelBw.Default*) → float

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:OBWLimit:CAGGregation:CBANdwidth<Band1>:CBANdwidth
↪<Band2>:CBANdwidth<Band3>
value: float or bool = driver.configure.multiEval.limit.seMask.obwLimit.
↪carrierAggregation.channelBw1st.channelBw2nd.channelBw3rd.get(firstChannelBw␣
↪= repcap.FirstChannelBw.Default, secondChannelBw = repcap.SecondChannelBw.
↪Default, thirdChannelBw = repcap.ThirdChannelBw.Default)
```

Defines an upper limit for the occupied bandwidth. The settings are defined separately for each channel bandwidth combination, for three aggregated carriers. The following bandwidth combinations are supported: Example: For the first line in the figure, use ...:CBANdwidth200:CBANdwidth150:CBANdwidth100.

param firstChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw1st')

param secondChannelBw
optional repeated capability selector. Default value: Bw50 (settable in the interface 'ChannelBw2nd')

param thirdChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw3rd')

return
obw_limit: (float or boolean) numeric | ON | OFF Range: 0 MHz to 40 MHz, Unit: Hz ON | OFF enables or disables the limit check.

set(*obw_limit: float*, *firstChannelBw=FirstChannelBw.Default*, *secondChannelBw=SecondChannelBw.Default*, *thirdChannelBw=ThirdChannelBw.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:OBWLimit:CAGGregation:CBANdwidth<Band1>:CBANdwidth
↪<Band2>:CBANdwidth<Band3>
driver.configure.multiEval.limit.seMask.obwLimit.carrierAggregation.
↪channelBw1st.channelBw2nd.channelBw3rd.set(obw_limit = 1.0, firstChannelBw =␣
↪repcap.FirstChannelBw.Default, secondChannelBw = repcap.SecondChannelBw.
↪Default, thirdChannelBw = repcap.ThirdChannelBw.Default)
```

Defines an upper limit for the occupied bandwidth. The settings are defined separately for each channel bandwidth combination, for three aggregated carriers. The following bandwidth combinations are supported: Example: For the first line in the figure, use ...:CBANdwidth200:CBANdwidth150:CBANdwidth100.

param obw_limit
(float or boolean) numeric | ON | OFF Range: 0 MHz to 40 MHz, Unit: Hz ON | OFF enables or disables the limit check.

param firstChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw1st')

param secondChannelBw
optional repeated capability selector. Default value: Bw50 (settable in the interface 'ChannelBw2nd')

param thirdChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw3rd')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.seMask.obwLimit.carrierAggregation.
↪channelBw1st.channelBw2nd.channelBw3rd.clone()
```

6.1.4.3.5.21 ChannelBw<ChannelBw>

RepCap Settings

```
# Range: Bw14 .. Bw200
rc = driver.configure.multiEval.limit.seMask.obwLimit.channelBw.repcap_channelBw_get()
driver.configure.multiEval.limit.seMask.obwLimit.channelBw.repcap_channelBw_set(repcap.
↪ChannelBw.Bw14)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:OBWLimit:CBANdwidth<Band>
```

class ChannelBwCls

ChannelBw commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: ChannelBw, default value after init: ChannelBw.Bw14

get(*channelBw=ChannelBw.Default*) → float

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:OBWLimit:CBANdwidth<Band>
value: float or bool = driver.configure.multiEval.limit.seMask.obwLimit.
↪channelBw.get(channelBw = repcap.ChannelBw.Default)
```

Defines an upper limit for the occupied bandwidth, depending on the channel bandwidth.

param channelBw
optional repeated capability selector. Default value: Bw14 (settable in the interface 'ChannelBw')

return
obw_limit: (float or boolean) numeric | ON | OFF Range: 0 MHz to 40 MHz, Unit: Hz ON | OFF enables or disables the limit check.

set(*obw_limit: float*, *channelBw=ChannelBw.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:OBWLimit:CBANdwidth<Band>
driver.configure.multiEval.limit.seMask.obwLimit.channelBw.set(obw_limit = 1.0,␣
↪channelBw = repcap.ChannelBw.Default)
```

Defines an upper limit for the occupied bandwidth, depending on the channel bandwidth.

param obw_limit
(float or boolean) numeric | ON | OFF Range: 0 MHz to 40 MHz, Unit: Hz ON | OFF enables or disables the limit check.

param channelBw
optional repeated capability selector. Default value: Bw14 (settable in the interface 'ChannelBw')

Cloning the Group

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

6.1.4.4 ListPy

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:OSINdex
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:PLCMode
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:CMODe
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:NCONnections
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST
```

class ListPyCls

ListPy commands group definition. 25 total commands, 3 Subgroups, 5 group commands

get_cmode() → ParameterSetMode

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:CMODe
value: enums.ParameterSetMode = driver.configure.multiEval.listPy.get_cmode()
```

No command help available

return
connector_mode: No help available

get_nconnections() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:NCONnections
value: int = driver.configure.multiEval.listPy.get_nconnections()
```

No command help available

return
no_of_connections: No help available

get_os_index() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:OSINdex
value: int or bool = driver.configure.multiEval.listPy.get_os_index()
```

Selects the number of the segment to be displayed in offline mode. The index refers to the range of measured segments, see method RsCmwLteMeas.Configure.MultiEval.ListPy.Lrange.set. Setting a value also enables the offline mode.

return
offline_seg_index: (integer or boolean) numeric | OFF Range: 1 to number of measured segments OFF disables the offline mode.

get_plc_mode() → ParameterSetMode

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:PLCMode
value: enums.ParameterSetMode = driver.configure.multiEval.listPy.get_plc_mode()
```

Selects which physical cell ID setting is used for list mode measurements.

return
plc_id_mode: GLOBal | LIST GLOBal The global setting is used for all segments, see method RsCmwLteMeas.Configure.MultiEval.Cc.PlcId.set. LIST The cell ID is configured per segment, see method RsCmwLteMeas.Configure.MultiEval.ListPy.Segment.PlcId.set.

get_value() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST
value: bool = driver.configure.multiEval.listPy.get_value()
```

Enables or disables the list mode.

return
enable: OFF | ON OFF: Disable list mode. ON: Enable list mode.

set_cmode(*connector_mode: ParameterSetMode*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:CMODe
driver.configure.multiEval.listPy.set_cmode(connector_mode = enums.
↪ParameterSetMode.GLOBal)
```

No command help available

param connector_mode
No help available

set_nconnections(*no_of_connections: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:NCONnections
driver.configure.multiEval.listPy.set_nconnections(no_of_connections = 1)
```

No command help available

param no_of_connections
No help available

set_os_index(*offline_seg_index: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:OSINdex
driver.configure.multiEval.listPy.set_os_index(offline_seg_index = 1)
```

Selects the number of the segment to be displayed in offline mode. The index refers to the range of measured segments, see method RsCmwLteMeas.Configure.MultiEval.ListPy.Lrange.set. Setting a value also enables the offline mode.

param offline_seg_index
(integer or boolean) numeric | OFF Range: 1 to number of measured segments OFF disables the offline mode.

set_plc_mode(*plc_id_mode: ParameterSetMode*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:PLCMode
driver.configure.multiEval.listPy.set_plc_mode(plc_id_mode = enums.
↪ParameterSetMode.GLOBal)
```

Selects which physical cell ID setting is used for list mode measurements.

param plc_id_mode
GLOBal | LIST GLOBal The global setting is used for all segments, see method RsCmwLteMeas.Configure.MultiEval.Cc.PlcId.set. LIST The cell ID is configured per segment, see method RsCmwLteMeas.Configure.MultiEval.ListPy.Segment.PlcId.set.

set_value(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST
driver.configure.multiEval.listPy.set_value(enable = False)
```

Enables or disables the list mode.

param enable
OFF | ON OFF: Disable list mode. ON: Enable list mode.

Cloning the Group

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

Subgroups

6.1.4.4.1 Lrange

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:LRANge
```

class LrangeCls

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

class LrangeStruct

Response structure. Fields:

- Start_Index: int: numeric First measured segment in the range of configured segments Range: 1 to 2000
- Nr_Segments: int: numeric Number of measured segments Range: 1 to 1000

get() → LrangeStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:LRANge
value: LrangeStruct = driver.configure.multiEval.listPy.lrange.get()
```

Select a range of measured segments. The segments must be configured using method RsCmwLteMeas.Configure.MultiEval.ListPy. Segment.Setup.set.

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

set(*start_index: int*, *nr_segments: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:LRANge
driver.configure.multiEval.listPy.lrange.set(start_index = 1, nr_segments = 1)
```

Select a range of measured segments. The segments must be configured using method RsCmwLteMeas.Configure.MultiEval.ListPy. Segment.Setup.set.

param start_index
numeric First measured segment in the range of configured segments Range: 1 to 2000

param nr_segments
numeric Number of measured segments Range: 1 to 1000

6.1.4.4.2 Segment

RepCap Settings

```
# Range: Nr1 .. Nr128
rc = driver.configure.multiEval.listPy.segment.repcap_segment_get()
driver.configure.multiEval.listPy.segment.repcap_segment_set(repcap.Segment.Nr1)
```

class SegmentCls

Segment commands group definition. 18 total commands, 15 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.configure.multiEval.listPy.segment.clone()
```

Subgroups

6.1.4.4.2.1 Aclr

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:ACLR
```

class AclrCls

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

class AclrStruct

Response structure. Fields:

- Aclr_Statistics: int: integer Statistical length in slots Range: 1 to 1000

- Aclr_Enable: bool: OFF | ON Enable or disable the measurement of ACLR results. ON: ACLR results are measured according to the other enable flags in this command. ACLR results for which there is no explicit enable flag are also measured (e.g. power in assigned E-UTRA channel) . OFF: No ACLR results at all are measured. The other enable flags in this command are ignored.
- Utra_1_Enable: bool: OFF | ON Enable or disable evaluation of first adjacent UTRA channels.
- Utra_2_Enable: bool: OFF | ON Enable or disable evaluation of second adjacent UTRA channels.
- Eutra_Enable: bool: OFF | ON Enable or disable evaluation of first adjacent E-UTRA channels.

get(*segment=Segment.Default*) → AclrStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:ACLR
value: AclrStruct = driver.configure.multiEval.listPy.segment.aclr.get(segment␣
↪= repcap.Segment.Default)
```

Defines settings for ACLR measurements in list mode for segment <no>.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

set(*aclr_statistics: int*, *aclr_enable: bool*, *utra_1_enable: bool*, *utra_2_enable: bool*, *eutra_enable: bool*, *segment=Segment.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:ACLR
driver.configure.multiEval.listPy.segment.aclr.set(aclr_statistics = 1, aclr_
↪enable = False, utra_1_enable = False, utra_2_enable = False, eutra_enable =␣
↪False, segment = repcap.Segment.Default)
```

Defines settings for ACLR measurements in list mode for segment <no>.

param aclr_statistics
integer Statistical length in slots Range: 1 to 1000

param aclr_enable
OFF | ON Enable or disable the measurement of ACLR results. ON: ACLR results are measured according to the other enable flags in this command. ACLR results for which there is no explicit enable flag are also measured (e.g. power in assigned E-UTRA channel) . OFF: No ACLR results at all are measured. The other enable flags in this command are ignored.

param utra_1_enable
OFF | ON Enable or disable evaluation of first adjacent UTRA channels.

param utra_2_enable
OFF | ON Enable or disable evaluation of second adjacent UTRA channels.

param eutra_enable
OFF | ON Enable or disable evaluation of first adjacent E-UTRA channels.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

6.1.4.4.2.2 CarrierAggregation

class CarrierAggregationCls

CarrierAggregation commands group definition. 3 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.1.4.4.2.3 AcSpacing

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:CAGGregation:ACSPacing
```

class AcSpacingCls

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

set(*segment=Segment.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:CAGGregation:ACSPacing
driver.configure.multiEval.listPy.segment.carrierAggregation.acSpacing.
↪set(segment = repcap.Segment.Default)
```

Adjusts the component carrier frequencies in segment <no>, so that the carriers are aggregated contiguously.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

set_with_opc(*segment=Segment.Default*, *opc_timeout_ms: int = -1*) → None

6.1.4.4.2.4 Mcarrier

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:CAGGregation:MCARrier
```

class McarrierCls

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

get(*segment=Segment.Default*) → MeasCarrier

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:CAGGregation:MCARrier
value: enums.MeasCarrier = driver.configure.multiEval.listPy.segment.
↪carrierAggregation.mcarrier.get(segment = repcap.Segment.Default)
```

No command help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
meas_carrier: No help available

set(*meas_carrier: MeasCarrier*, *segment=Segment.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:CAGGregation:MCARrier
driver.configure.multiEval.listPy.segment.carrierAggregation.mcarrier.set(meas_
↪carrier = enums.MeasCarrier.PCC, segment = repcap.Segment.Default)
```

No command help available

param meas_carrier
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

Cloning the Group

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

Subgroups

6.1.4.4.2.5 Enhanced

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:CAGGregation:MCARrier:ENHanced
```

class EnhancedCls

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

get(*segment=Segment.Default*) → MeasCarrierEnhanced

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:CAGGregation:MCARrier:ENHanced
value: enums.MeasCarrierEnhanced = driver.configure.multiEval.listPy.segment.
↪carrierAggregation.mcarrier.enhanced.get(segment = repcap.Segment.Default)
```

Selects a component carrier for single-carrier measurements in segment <no>.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
meas_carrier: CC1 | CC2 | CC3 | CC4

set(*meas_carrier: MeasCarrierEnhanced*, *segment=Segment.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:CAGGregation:MCARrier:ENHanced
driver.configure.multiEval.listPy.segment.carrierAggregation.mcarrier.enhanced.
↪set(meas_carrier = enums.MeasCarrierEnhanced.CC1, segment = repcap.Segment.
↪Default)
```

Selects a component carrier for single-carrier measurements in segment <no>.

param meas_carrier
CC1 | CC2 | CC3 | CC4

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

6.1.4.4.2.6 Cc<CarrierComponent>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.configure.multiEval.listPy.segment.cc.repcap_carrierComponent_get()
driver.configure.multiEval.listPy.segment.cc.repcap_carrierComponent_set(repcap.
↪CarrierComponent.Nr1)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:CC<c>
```

class CcCls

Cc commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: CarrierComponent, default value after init: CarrierComponent.Nr1

class CcStruct

Response structure. Fields:

- Frequency: float: numeric Center frequency of the component carrier, used in the segment For the supported range, see 'Frequency ranges'. Unit: Hz
- Ch_Bandwidth: enums.ChannelBandwidth: B014 | B030 | B050 | B100 | B150 | B200 Channel bandwidth of the component carrier, used in the segment B014: 1.4 MHz B030: 3 MHz B050: 5 MHz B100: 10 MHz B150: 15 MHz B200: 20 MHz

get(*segment=Segment.Default*, *carrierComponent=CarrierComponent.Default*) → CcStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:CC<c>
value: CcStruct = driver.configure.multiEval.listPy.segment.cc.get(segment =␣
↪repcap.Segment.Default, carrierComponent = repcap.CarrierComponent.Default)
```

Defines carrier-specific analyzer settings for component carrier CC<c>, in segment <no>. This command is only relevant for carrier aggregation.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

set(*frequency: float*, *ch_bandwidth: ChannelBandwidth*, *segment=Segment.Default*, *carrierComponent=CarrierComponent.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:CC<c>
driver.configure.multiEval.listPy.segment.cc.set(frequency = 1.0, ch_bandwidth␣
↪= enums.ChannelBandwidth.B014, segment = repcap.Segment.Default,␣
↪carrierComponent = repcap.CarrierComponent.Default)
```

Defines carrier-specific analyzer settings for component carrier CC<c>, in segment <no>. This command is only relevant for carrier aggregation.

param frequency
numeric Center frequency of the component carrier, used in the segment For the supported range, see 'Frequency ranges'. Unit: Hz

param ch_bandwidth
B014 | B030 | B050 | B100 | B150 | B200 Channel bandwidth of the component carrier, used in the segment B014: 1.4 MHz B030: 3 MHz B050: 5 MHz B100: 10 MHz B150: 15 MHz B200: 20 MHz

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

Cloning the Group

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

6.1.4.4.2.7 Cidx

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:CIDX
```

class CidxCls

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

get(*segment=Segment.Default*) → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:CIDX
value: int = driver.configure.multiEval.listPy.segment.cidx.get(segment =␣
↪repcap.Segment.Default)
```

No command help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
connection_index: No help available

set(*connection_index: int*, *segment=Segment.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:CIDX
driver.configure.multiEval.listPy.segment.cidx.set(connection_index = 1,␣
↪segment = repcap.Segment.Default)
```

No command help available

param connection_index
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

6.1.4.4.2.8 Emtc

class EmtcCls

Emtc 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.multiEval.listPy.segment.emtc.clone()
```

Subgroups

6.1.4.4.2.9 Nband

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:EMTC:NBANd
```

class NbandCls

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

get(*segment=Segment.Default*) → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:EMTC:NBANd
value: int = driver.configure.multiEval.listPy.segment.emtc.nband.get(segment =␣
↪repcap.Segment.Default)
```

Selects the eMTC narrowband for segment <no>.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
number: numeric The maximum depends on the channel BW, see 'RB allocation, narrowbands and widebands for eMTC'. Range: 0 to 15

set(*number: int*, *segment=Segment.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:EMTC:NBANd
driver.configure.multiEval.listPy.segment.emtc.nband.set(number = 1, segment =␣
↪repcap.Segment.Default)
```

Selects the eMTC narrowband for segment <no>.

param number
numeric The maximum depends on the channel BW, see 'RB allocation, narrowbands and widebands for eMTC'. Range: 0 to 15

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

6.1.4.4.2.10 Modulation

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:MODulation
```

class ModulationCls

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

class ModulationStruct

Structure for setting input parameters. Fields:

- Mod_Statistics: int: integer Statistical length in slots. Range: 1 to 1000
- Modenable: bool: OFF | ON Enable or disable the measurement of modulation results. ON: Modulation results are measured according to the other enable flags in this command. Modulation results for which there is no explicit enable flag are also measured (e.g. I/Q offset, frequency error and timing error) . OFF: No modulation results at all are measured. The other enable flags in this command are ignored.
- Evm_Enable: bool: OFF | ON Enable or disable measurement of EVM.
- Mag_Error_Enable: bool: OFF | ON Enable or disable measurement of magnitude error.
- Phase_Err_Enable: bool: OFF | ON Enable or disable measurement of phase error.
- Ib_Eenable: bool: OFF | ON Enable or disable measurement of inband emissions.
- Eq_Sp_Flat_Enable: bool: OFF | ON Enable or disable measurement of equalizer spectrum flatness results.
- Mod_Scheme: enums.ModScheme: AUTO | QPSK | Q16 | Q64 | Q256 Modulation scheme used by the LTE uplink signal. AUTO: automatic detection QPSK: QPSK Q16: 16-QAM Q64: 64-QAM Q256: 256-QAM

get(*segment=Segment.Default*) → ModulationStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:MODulation
value: ModulationStruct = driver.configure.multiEval.listPy.segment.modulation.
↪get(segment = repcap.Segment.Default)
```

Defines settings for modulation measurements in list mode for segment <no>.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

set(*structure: ModulationStruct*, *segment=Segment.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:MODulation
structure = driver.configure.multiEval.listPy.segment.modulation.
↪ModulationStruct()
structure.Mod_Statistics: int = 1
structure.Modenable: bool = False
structure.Evm_Enable: bool = False
structure.Mag_Error_Enable: bool = False
structure.Phase_Err_Enable: bool = False
structure.Ib_Eenable: bool = False
structure.Eq_Sp_Flat_Enable: bool = False
structure.Mod_Scheme: enums.ModScheme = enums.ModScheme.AUTO
driver.configure.multiEval.listPy.segment.modulation.set(structure, segment =␣
↪repcap.Segment.Default)
```

Defines settings for modulation measurements in list mode for segment <no>.

param structure
for set value, see the help for ModulationStruct structure arguments.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

6.1.4.4.2.11 PlcId

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:PLCid
```

class PlcIdCls

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

get(*segment=Segment.Default*) → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:PLCid
value: int = driver.configure.multiEval.listPy.segment.plcId.get(segment =␣
↪repcap.Segment.Default)
```

Specifies the physical cell ID for segment <no>. See also method RsCmwLteMeas.Configure.MultiEval.ListPy.plcMode.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
phys_layer_cell_id: integer Range: 0 to 503

set(*phys_layer_cell_id: int*, *segment=Segment.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:PLCid
driver.configure.multiEval.listPy.segment.plcId.set(phys_layer_cell_id = 1,␣
↪segment = repcap.Segment.Default)
```

Specifies the physical cell ID for segment <no>. See also method RsCmwLteMeas.Configure.MultiEval.ListPy.plcMode.

param phys_layer_cell_id
integer Range: 0 to 503

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

6.1.4.4.2.12 Pmonitor

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:PMONitor
```

class PmonitorCls

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

get(*segment=Segment.Default*) → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:PMONitor
value: bool = driver.configure.multiEval.listPy.segment.pmonitor.get(segment =␣
↪repcap.Segment.Default)
```

Enables or disables the measurement of power monitor results (power of one carrier) for segment <no>.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
enable: OFF | ON

set(*enable: bool*, *segment=Segment.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:PMONitor
driver.configure.multiEval.listPy.segment.pmonitor.set(enable = False, segment␣
↪= repcap.Segment.Default)
```

Enables or disables the measurement of power monitor results (power of one carrier) for segment <no>.

param enable
OFF | ON

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

6.1.4.4.2.13 Power

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer
```

class PowerCls

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

class PowerStruct

Response structure. Fields:

- Power_Statistics: int: integer Statistical length in subframes Range: 1 to 1000
- Power_Enable: bool: OFF | ON Enables or disables the measurement of the total TX power.

get(*segment=Segment.Default*) → PowerStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer
value: PowerStruct = driver.configure.multiEval.listPy.segment.power.
↪get(segment = repcap.Segment.Default)
```

Defines settings for the measurement of the total TX power of all carriers for segment <no>.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

set(*power_statistics: int*, *power_enable: bool*, *segment=Segment.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer
driver.configure.multiEval.listPy.segment.power.set(power_statistics = 1, power_
↪enable = False, segment = repcap.Segment.Default)
```

Defines settings for the measurement of the total TX power of all carriers for segment <no>.

param power_statistics
integer Statistical length in subframes Range: 1 to 1000

param power_enable
OFF | ON Enables or disables the measurement of the total TX power.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

6.1.4.4.2.14 RbAllocation

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:RBALlocation
```

class RbAllocationCls

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

class RbAllocationStruct

Response structure. Fields:

- Auto: bool: OFF | ON OFF: manual definition via NoRB and Offset ON: automatic detection of RB allocation
- No_Rb: int: integer Number of allocated resource blocks in each measured slot Range: see table below
- Offset: int: integer Offset of first allocated resource block from edge of allocated UL transmission bandwidth Range: 0 to max(NoRB) - NoRB

get(*segment=Segment.Default*) → RbAllocationStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:RBALlocation
value: RbAllocationStruct = driver.configure.multiEval.listPy.segment.
↪rbAllocation.get(segment = repcap.Segment.Default)
```

Allows you to define the uplink resource block allocation manually for segment <no>. By default, the RB allocation is detected automatically.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

set(*auto: bool*, *no_rb: int*, *offset: int*, *segment=Segment.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:RBALlocation
driver.configure.multiEval.listPy.segment.rbAllocation.set(auto = False, no_rb␣
↪= 1, offset = 1, segment = repcap.Segment.Default)
```

Allows you to define the uplink resource block allocation manually for segment <no>. By default, the RB allocation is detected automatically.

param auto
OFF | ON OFF: manual definition via NoRB and Offset ON: automatic detection of RB allocation

param no_rb
integer Number of allocated resource blocks in each measured slot Range: see table below

param offset
integer Offset of first allocated resource block from edge of allocated UL transmission bandwidth Range: 0 to max(NoRB) - NoRB

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

Cloning the Group

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

Subgroups

6.1.4.4.2.15 Sidelink

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:RBALlocation:SIDelink
```

class SidelinkCls

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

class SidelinkStruct

Response structure. Fields:

- Auto: bool: OFF | ON OFF: manual definition via the other settings ON: automatic detection of RB allocation
- No_Rb_Pssch: int: integer Number of allocated RBs for the PSSCH in each measured slot
- Offset_Pssch: int: integer Offset of the first allocated PSSCH resource block
- Offset_Pscch: int: integer Offset of the first allocated PSCCH resource block

get(*segment=Segment.Default*) → SidelinkStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:RBALlocation:SIDelink
value: SidelinkStruct = driver.configure.multiEval.listPy.segment.rbAllocation.
↪sidelink.get(segment = repcap.Segment.Default)
```

Allows you to define the sidelink resource block allocation manually for segment <no>. By default, the RB allocation is detected automatically. Most allowed input ranges depend on other settings, see 'Sidelink resource block allocation'.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

set(*auto: bool*, *no_rb_pssch: int*, *offset_pssch: int*, *offset_pscch: int*, *segment=Segment.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:RBALlocation:SIDelink
driver.configure.multiEval.listPy.segment.rbAllocation.sidelink.set(auto =␣
↪False, no_rb_pssch = 1, offset_pssch = 1, offset_pscch = 1, segment = repcap.
↪Segment.Default)
```

Allows you to define the sidelink resource block allocation manually for segment <no>. By default, the RB allocation is detected automatically. Most allowed input ranges depend on other settings, see 'Sidelink resource block allocation'.

param auto
OFF | ON OFF: manual definition via the other settings ON: automatic detection of RB allocation

param no_rb_pssch
integer Number of allocated RBs for the PSSCH in each measured slot

param offset_pssch
integer Offset of the first allocated PSSCH resource block

param offset_pscch
integer Offset of the first allocated PSCCH resource block

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

6.1.4.4.2.16 Scc<SecondaryCC>

RepCap Settings

```
# Range: CC1 .. CC7
rc = driver.configure.multiEval.listPy.segment.scc.repcap_secondaryCC_get()
driver.configure.multiEval.listPy.segment.scc.repcap_secondaryCC_set(repcap.SecondaryCC.
↪CC1)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SCC<c>
```

class SccCls

Scc commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: SecondaryCC, default value after init: SecondaryCC.CC1

class SccStruct

Response structure. Fields:

- Frequency: float: No parameter help available
- Ch_Bandwidth: enums.ChannelBandwidth: No parameter help available

get(*segment=Segment.Default*, *secondaryCC=SecondaryCC.Default*) → SccStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SCC<c>
value: SccStruct = driver.configure.multiEval.listPy.segment.scc.get(segment =␣
↪repcap.Segment.Default, secondaryCC = repcap.SecondaryCC.Default)
```

No command help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

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

set(*frequency: float*, *ch_bandwidth: ChannelBandwidth*, *segment=Segment.Default*, *secondaryCC=SecondaryCC.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SCC<c>
driver.configure.multiEval.listPy.segment.scc.set(frequency = 1.0, ch_bandwidth␣
↪= enums.ChannelBandwidth.B014, segment = repcap.Segment.Default, secondaryCC␣
↪= repcap.SecondaryCC.Default)
```

No command help available

param frequency
No help available

param ch_bandwidth
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

Cloning the Group

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

6.1.4.4.2.17 SeMask

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SEMask
```

class SeMaskCls

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

class SeMaskStruct

Response structure. Fields:

- Sem_Statistics: int: integer Statistical length in slots. Range: 1 to 1000
- Se_Enable: bool: OFF | ON Enable or disable the measurement of spectrum emission results. ON: Spectrum emission results are measured according to the other enable flags in this command. Results for which there is no explicit enable flag are also measured. OFF: No spectrum emission results at all are measured. The other enable flags in this command are ignored.
- Obw_Enable: bool: OFF | ON Enable or disable measurement of occupied bandwidth.

- Sem_Enable: bool: OFF | ON Enable or disable measurement of spectrum emission trace and margin results.

get(*segment=Segment.Default*) → SeMaskStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SEMask
value: SeMaskStruct = driver.configure.multiEval.listPy.segment.seMask.
↪get(segment = repcap.Segment.Default)
```

Defines settings for spectrum emission measurements in list mode for segment <no>.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

set(*sem_statistics: int*, *se_enable: bool*, *obw_enable: bool*, *sem_enable: bool*, *segment=Segment.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SEMask
driver.configure.multiEval.listPy.segment.seMask.set(sem_statistics = 1, se_
↪enable = False, obw_enable = False, sem_enable = False, segment = repcap.
↪Segment.Default)
```

Defines settings for spectrum emission measurements in list mode for segment <no>.

param sem_statistics
integer Statistical length in slots. Range: 1 to 1000

param se_enable
OFF | ON Enable or disable the measurement of spectrum emission results. ON: Spectrum emission results are measured according to the other enable flags in this command. Results for which there is no explicit enable flag are also measured. OFF: No spectrum emission results at all are measured. The other enable flags in this command are ignored.

param obw_enable
OFF | ON Enable or disable measurement of occupied bandwidth.

param sem_enable
OFF | ON Enable or disable measurement of spectrum emission trace and margin results.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

6.1.4.4.2.18 Setup

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SETup
```

class SetupCls

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

class SetupStruct

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

- Segment_Length: int: integer Number of subframes in the segment Range: 1 to 2000
- Level: float: numeric Expected nominal power in the segment. The range can be calculated as follows: Range (Expected Nominal Power) = Range (Input Power) + External Attenuation - User Margin The input power range is stated in the data sheet. Unit: dBm
- Duplex_Mode: enums.DuplexMode: FDD | TDD Duplex mode used in the segment
- Band: enums.Band: FDD UL: OB1 | … | OB28 | OB30 | OB31 | OB65 | OB66 | OB68 | OB70 | … | OB74 | OB85 | OB87 | OB88 TDD UL: OB33 | … | OB45 | OB48 | OB50 | … | OB53 | OB250 Sidelink: OB47 Operating band used in the segment
- Frequency: float: numeric Center frequency of CC1 used in the segment For the supported range, see 'Frequency ranges'. Unit: Hz
- Ch_Bandwidth: enums.ChannelBandwidth: B014 | B030 | B050 | B100 | B150 | B200 Channel bandwidth of CC1 used in the segment B014: 1.4 MHz B030: 3 MHz B050: 5 MHz B100: 10 MHz B150: 15 MHz B200: 20 MHz
- Cyclic_Prefix: enums.CyclicPrefix: NORMal | EXTended Type of cyclic prefix used in the segment
- Channel_Type: enums.SegmentChannelTypeExtended: AUTO | PUSCh | PUCCh | PSSCh | PSCCh Channel type to be measured in the segment (AUTO for automatic detection) Uplink: AUTO, PUSCh, PUCCh Sidelink: PSSCh, PSCCh
- Retrigger_Flag: enums.RetriggerFlag: OFF | ON | IFPower | IFPNarrow Specifies whether the measurement waits for a trigger event before measuring the segment, or not. The retrigger flag is ignored for trigger mode ONCE and evaluated for trigger mode SEGMent, see [CMDLINKRESOLVED Trigger.MultiEval.ListPy#Mode CMDLINKRESOLVED]. OFF Measure the segment without retrigger. For the first segment, the value OFF is interpreted as ON. ON Wait for a trigger event from the trigger source configured via [CMDLINKRESOLVED Trigger.MultiEval#Source CMDLINKRESOLVED]. IFPower Wait for a trigger event from the trigger source 'IF Power'. The trigger evaluation bandwidth is 160 MHz. IFPNarrowband Wait for a trigger event from the trigger source 'IF Power'. The trigger evaluation bandwidth is configured via [CMDLINKRESOLVED Trigger.MultiEval.ListPy#Nbandwidth CMDLINKRESOLVED].
- Evaluat_Offset: int: integer Number of subframes at the beginning of the segment that are not evaluated Range: 0 to 1000
- Network_Sig_Value: enums.NetworkSigValueNoCarrAggr: Optional setting parameter. NS01 | … | NS288 Network signaled value to be used for the segment

get(*segment=Segment.Default*) → SetupStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SETup
value: SetupStruct = driver.configure.multiEval.listPy.segment.setup.
↪get(segment = repcap.Segment.Default)
```

Defines the length and analyzer settings of segment <no>. This command must be sent for all segments to be measured (method RsCmwLteMeas.Configure.MultiEval.ListPy.Lrange.set) . For uplink signals with TDD mode, see also method RsCmwLteMeas.Configure.MultiEval.ListPy.Segment.Tdd.set. For carrier-specific settings for carrier aggregation, see CONFigure:LTE:MEAS<i>:MEValuation:LIST:SEGMent<no>:CC<c>.

param segment
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*, *segment=Segment.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SETup
structure = driver.configure.multiEval.listPy.segment.setup.SetupStruct()
structure.Segment_Length: int = 1
structure.Level: float = 1.0
structure.Duplex_Mode: enums.DuplexMode = enums.DuplexMode.FDD
structure.Band: enums.Band = enums.Band.OB1
structure.Frequency: float = 1.0
structure.Ch_Bandwidth: enums.ChannelBandwidth = enums.ChannelBandwidth.B014
structure.Cyclic_Prefix: enums.CyclicPrefix = enums.CyclicPrefix.EXTended
structure.Channel_Type: enums.SegmentChannelTypeExtended = enums.
↪SegmentChannelTypeExtended.AUTO
structure.Retrigger_Flag: enums.RetriggerFlag = enums.RetriggerFlag.IFPNarrow
structure.Evaluat_Offset: int = 1
structure.Network_Sig_Value: enums.NetworkSigValueNoCarrAggr = enums.
↪NetworkSigValueNoCarrAggr.NS01
driver.configure.multiEval.listPy.segment.setup.set(structure, segment = repcap.
↪Segment.Default)
```

Defines the length and analyzer settings of segment <no>. This command must be sent for all segments to be measured (method RsCmwLteMeas.Configure.MultiEval.ListPy.Lrange.set) . For uplink signals with TDD mode, see also method RsCmwLteMeas.Configure.MultiEval.ListPy.Segment.Tdd.set. For carrier-specific settings for carrier aggregation, see CONFigure:LTE:MEAS<i>:MEValuation:LIST:SEGMent<no>:CC<c>.

param structure
for set value, see the help for SetupStruct structure arguments.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

6.1.4.4.2.19 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.multiEval.listPy.segment.singleCmw.clone()
```

Subgroups

6.1.4.4.2.20 Connector

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:CMWS:CONNector
```

class ConnectorCls

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

get(*segment=Segment.Default*) → CmwsConnector

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:CMWS:CONNector
value: enums.CmwsConnector = driver.configure.multiEval.listPy.segment.
↪singleCmw.connector.get(segment = repcap.Segment.Default)
```

No command help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
cmws_connector: No help available

set(*cmws_connector: CmwsConnector*, *segment=Segment.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:CMWS:CONNector
driver.configure.multiEval.listPy.segment.singleCmw.connector.set(cmws_
↪connector = enums.CmwsConnector.R11, segment = repcap.Segment.Default)
```

No command help available

param cmws_connector
No help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

6.1.4.4.2.21 Tdd

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:TDD
```

class TddCls

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

class TddStruct

Response structure. Fields:

- Uplink_Downlink: int: integer UL-DL configuration, defining the combination of uplink, downlink and special subframes within a radio frame Range: 0 to 6
- Special_Subframe: int: integer Special subframe configuration, defining the inner structure of special subframes Range: 0 to 8

get(*segment=Segment.Default*) → TddStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:TDD
value: TddStruct = driver.configure.multiEval.listPy.segment.tdd.get(segment =␣
↪repcap.Segment.Default)
```

Defines segment settings only relevant for uplink measurements with the duplex mode TDD. For general segment configuration, see method RsCmwLteMeas.Configure.MultiEval.ListPy.Segment.Setup.set.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

set(*uplink_downlink: int*, *special_subframe: int*, *segment=Segment.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:TDD
driver.configure.multiEval.listPy.segment.tdd.set(uplink_downlink = 1, special_
↪subframe = 1, segment = repcap.Segment.Default)
```

Defines segment settings only relevant for uplink measurements with the duplex mode TDD. For general segment configuration, see method RsCmwLteMeas.Configure.MultiEval.ListPy.Segment.Setup.set.

param uplink_downlink
integer UL-DL configuration, defining the combination of uplink, downlink and special subframes within a radio frame Range: 0 to 6

param special_subframe
integer Special subframe configuration, defining the inner structure of special subframes Range: 0 to 8

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

6.1.4.4.3 SingleCmw

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:CMWS:CMODe
```

class SingleCmwCls

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

get_cmode() → ParameterSetMode

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:CMWS:CMODe
value: enums.ParameterSetMode = driver.configure.multiEval.listPy.singleCmw.get_
↪cmode()
```

No command help available

return
connector_mode: No help available

set_cmode(*connector_mode: ParameterSetMode*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:CMWS:CMODe
driver.configure.multiEval.listPy.singleCmw.set_cmode(connector_mode = enums.
↪ParameterSetMode.GLOBal)
```

No command help available

param connector_mode
No help available

6.1.4.5 Modulation

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:MODulation:EQUalizer
CONFigure:LTE:MEASurement<Instance>:MEValuation:MODulation:MSCHeme
CONFigure:LTE:MEASurement<Instance>:MEValuation:MODulation:LLOCation
```

class ModulationCls

Modulation commands group definition. 9 total commands, 3 Subgroups, 3 group commands

get_equalizer() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:MODulation:EQUalizer
value: bool = driver.configure.multiEval.modulation.get_equalizer()
```

Enables or disables the post-FFT equalization step for the measurement of modulation results.

return
enable: OFF | ON

get_llocation() → LocalOscLocation

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:MODulation:LLOCation
value: enums.LocalOscLocation = driver.configure.multiEval.modulation.get_
↪llocation()
```

Specifies the UE transmitter architecture (local oscillator location) used for eMTC.

return
value: CN | CCB CN: Center of narrowband/wideband CCB: Center of channel bandwidth

get_mscheme() → ModScheme

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:MODulation:MSCHeme
value: enums.ModScheme = driver.configure.multiEval.modulation.get_mscheme()
```

Selects the modulation scheme used by the measured signal.

return
mod_scheme: AUTO | QPSK | Q16 | Q64 | Q256 Auto-detection, QPSK, 16-QAM, 64-QAM, 256-QAM

set_equalizer(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:MODulation:EQUalizer
driver.configure.multiEval.modulation.set_equalizer(enable = False)
```

Enables or disables the post-FFT equalization step for the measurement of modulation results.

param enable
OFF | ON

set_llocation(*value: LocalOscLocation*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:MODulation:LLOCation
driver.configure.multiEval.modulation.set_llocation(value = enums.
↪LocalOscLocation.CCB)
```

Specifies the UE transmitter architecture (local oscillator location) used for eMTC.

param value
CN | CCB CN: Center of narrowband/wideband CCB: Center of channel bandwidth

set_mscheme(*mod_scheme: ModScheme*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:MODulation:MSCHeme
driver.configure.multiEval.modulation.set_mscheme(mod_scheme = enums.ModScheme.
↪AUTO)
```

Selects the modulation scheme used by the measured signal.

param mod_scheme
AUTO | QPSK | Q16 | Q64 | Q256 Auto-detection, QPSK, 16-QAM, 64-QAM, 256-QAM

Cloning the Group

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

Subgroups

6.1.4.5.1 CarrierAggregation

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:MODulation:CAGGregation:LLOCation
```

class CarrierAggregationCls

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

get_llocation() → CarrAggrLocalOscLocation

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:MODulation:CAGGregation:LLOCation
value: enums.CarrAggrLocalOscLocation = driver.configure.multiEval.modulation.
↪carrierAggregation.get_llocation()
```

Specifies the UE transmitter architecture (local oscillator location) used for contiguous carrier aggregation.

return
value: CACB | CECC CACB: Center of aggregated channel bandwidth CECC: Center of each component carrier

set_llocation(*value: CarrAggrLocalOscLocation*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:MODulation:CAGGregation:LLOCation
driver.configure.multiEval.modulation.carrierAggregation.set_llocation(value =␣
↪enums.CarrAggrLocalOscLocation.AUTO)
```

Specifies the UE transmitter architecture (local oscillator location) used for contiguous carrier aggregation.

param value
CACB | CECC CACB: Center of aggregated channel bandwidth CECC: Center of each component carrier

6.1.4.5.2 EePeriods

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:MODulation:EEPeriods:PUCCh
```

class EePeriodsCls

EePeriods commands group definition. 3 total commands, 1 Subgroups, 1 group commands

get_pucch() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:MODulation:EEPeriods:PUCCh
value: bool = driver.configure.multiEval.modulation.eePeriods.get_pucch()
```

Enables or disables EVM exclusion periods for slots with detected channel type 'PUCCH'. If enabled, the first and the last SC-FDMA symbol of each slot is excluded from the calculation of EVM, magnitude error and phase error single value results. If the last symbol of a slot is already excluded because SRS signals are allowed, the second but last symbol is also excluded.

return
pucch: OFF | ON

set_pucch(*pucch: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:MODulation:EEPeriods:PUCCh
driver.configure.multiEval.modulation.eePeriods.set_pucch(pucch = False)
```

Enables or disables EVM exclusion periods for slots with detected channel type 'PUCCH'. If enabled, the first and the last SC-FDMA symbol of each slot is excluded from the calculation of EVM, magnitude error and phase error single value results. If the last symbol of a slot is already excluded because SRS signals are allowed, the second but last symbol is also excluded.

param pucch
OFF | ON

Cloning the Group

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

Subgroups

6.1.4.5.2.1 Pusch

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:MODulation:EEPeriods:PUSCh:LEADing
CONFigure:LTE:MEASurement<Instance>:MEValuation:MODulation:EEPeriods:PUSCh:LAGGing
```

class PuschCls

Pusch commands group definition. 2 total commands, 0 Subgroups, 2 group commands

get_lagging() → LaggingExclPeriod

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:MODulation:EEPeriods:PUSCh:LAGGing
value: enums.LaggingExclPeriod = driver.configure.multiEval.modulation.
↪eePeriods.pusch.get_lagging()
```

Specifies an EVM exclusion period at the end of each subframe (detected channel type 'PUSCH') ; if SRS signals are allowed, at the end of each shortened subframe. The specified period is excluded from the calculation of EVM, magnitude error and phase error results.

return
lagging: OFF | MS05 | MS25 OFF: no exclusion MS05: 5 s excluded MS25: 25 s excluded

get_leading() → LeadingExclPeriod

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:MODulation:EEPeriods:PUSCh:LEADing
value: enums.LeadingExclPeriod = driver.configure.multiEval.modulation.
↪eePeriods.pusch.get_leading()
```

Specifies an EVM exclusion period at the beginning of a subframe (detected channel type 'PUSCH') . The specified period is excluded from the calculation of EVM, magnitude error and phase error results.

return
leading: OFF | MS25 OFF: no exclusion MS25: 25 s excluded

set_lagging(*lagging: LaggingExclPeriod*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:MODulation:EEPeriods:PUSCh:LAGGing
driver.configure.multiEval.modulation.eePeriods.pusch.set_lagging(lagging =␣
↪enums.LaggingExclPeriod.MS05)
```

Specifies an EVM exclusion period at the end of each subframe (detected channel type 'PUSCH') ; if SRS signals are allowed, at the end of each shortened subframe. The specified period is excluded from the calculation of EVM, magnitude error and phase error results.

param lagging
OFF | MS05 | MS25 OFF: no exclusion MS05: 5 s excluded MS25: 25 s excluded

set_leading(*leading: LeadingExclPeriod*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:MODulation:EEPeriods:PUSCh:LEADing
driver.configure.multiEval.modulation.eePeriods.pusch.set_leading(leading =␣
↪enums.LeadingExclPeriod.MS25)
```

Specifies an EVM exclusion period at the beginning of a subframe (detected channel type 'PUSCH') . The specified period is excluded from the calculation of EVM, magnitude error and phase error results.

param leading
OFF | MS25 OFF: no exclusion MS25: 25 s excluded

6.1.4.5.3 EwLength

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:MODulation:EWLength
```

class EwLengthCls

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

class EwLengthStruct

Response structure. Fields:

- Length_Cp_Normal: List[int]: No parameter help available
- Length_Cp_Extended: List[int]: No parameter help available

get() → EwLengthStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:MODulation:EWLength
value: EwLengthStruct = driver.configure.multiEval.modulation.ewLength.get()
```

Specifies the EVM window length in samples for all channel bandwidths, depending on the cyclic prefix (CP) type.

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

set(*length_cp_normal: List[int]*, *length_cp_extended: List[int]*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:MODulation:EWLength
driver.configure.multiEval.modulation.ewLength.set(length_cp_normal = [1, 2, 3],
↪ length_cp_extended = [1, 2, 3])
```

Specifies the EVM window length in samples for all channel bandwidths, depending on the cyclic prefix (CP) type.

param length_cp_normal
No help available

param length_cp_extended
No help available

Cloning the Group

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

Subgroups

6.1.4.5.3.1 ChannelBw<ChannelBw>

RepCap Settings

```
# Range: Bw14 .. Bw200
rc = driver.configure.multiEval.modulation.ewLength.channelBw.repcap_channelBw_get()
driver.configure.multiEval.modulation.ewLength.channelBw.repcap_channelBw_set(repcap.
↪ChannelBw.Bw14)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:MODulation:EWLength:CBANdwidth<Band>
```

class ChannelBwCls

ChannelBw commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: ChannelBw, default value after init: ChannelBw.Bw14

class ChannelBwStruct

Response structure. Fields:

- Cyc_Prefix_Normal: int: integer Samples for normal CP Range: see below
- Cyc_Prefix_Extend: int: integer Samples for extended CP Range: see below

get(*channelBw=ChannelBw.Default*) → ChannelBwStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:MODulation:EWLength:CBANdwidth<Band>
value: ChannelBwStruct = driver.configure.multiEval.modulation.ewLength.
↪channelBw.get(channelBw = repcap.ChannelBw.Default)
```

Specifies the EVM window length in samples for a selected channel bandwidth, depending on the cyclic prefix (CP) type.

param channelBw
optional repeated capability selector. Default value: Bw14 (settable in the interface 'ChannelBw')

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

set(*cyc_prefix_normal: int*, *cyc_prefix_extend: int*, *channelBw=ChannelBw.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:MODulation:EWLength:CBANdwidth<Band>
driver.configure.multiEval.modulation.ewLength.channelBw.set(cyc_prefix_normal␣
↪= 1, cyc_prefix_extend = 1, channelBw = repcap.ChannelBw.Default)
```

Specifies the EVM window length in samples for a selected channel bandwidth, depending on the cyclic prefix (CP) type.

param cyc_prefix_normal
integer Samples for normal CP Range: see below

param cyc_prefix_extend
integer Samples for extended CP Range: see below

param channelBw
optional repeated capability selector. Default value: Bw14 (settable in the interface 'ChannelBw')

Cloning the Group

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

6.1.4.6 MsubFrames

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:MSUBframes
```

class MsubFramesCls

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

class MsubFramesStruct

Response structure. Fields:

- Sub_Frame_Offset: int: decimal Start of the measured subframe range relative to the trigger event. Range: 0 to 9
- Sub_Frame_Count: int: decimal Length of the measured subframe range. Range: 1 to 320
- Meas_Subframe: int: decimal Subframe containing the measured slots for modulation and spectrum results. Range: 0 to SubframeCount-1

get() → MsubFramesStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:MSUBframes
value: MsubFramesStruct = driver.configure.multiEval.msubFrames.get()
```

Configures the scope of the measurement, i.e. which subframes are measured.

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

set(*sub_frame_offset: int*, *sub_frame_count: int*, *meas_subframe: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:MSUBframes
driver.configure.multiEval.msubFrames.set(sub_frame_offset = 1, sub_frame_count␣
↪= 1, meas_subframe = 1)
```

Configures the scope of the measurement, i.e. which subframes are measured.

param sub_frame_offset
decimal Start of the measured subframe range relative to the trigger event. Range: 0 to 9

param sub_frame_count
decimal Length of the measured subframe range. Range: 1 to 320

param meas_subframe
decimal Subframe containing the measured slots for modulation and spectrum results. Range: 0 to SubframeCount-1

6.1.4.7 NsValue

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:NSValue:CAGGregation
CONFigure:LTE:MEASurement<Instance>:MEValuation:NSValue
```

class NsValueCls

NsValue commands group definition. 2 total commands, 0 Subgroups, 2 group commands

get_carrier_aggregation() → NetworkSigValue

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:NSValue:CAGGregation
value: enums.NetworkSigValue = driver.configure.multiEval.nsValue.get_carrier_
↪aggregation()
```

Selects the 'network signaled value' for measurements with carrier aggregation. For the combined signal path scenario, use CONFigure:LTE:SIGN<i>:CONNection:SCC<c>:ASEMission:CAGGregation.

return
value: NS01 | NS02 | NS03 | NS04 | NS05 | NS06 | NS07 | NS08 | NS09 | NS10 | NS11 | NS12 | NS13 | NS14 | NS15 | NS16 | NS17 | NS18 | NS19 | NS20 | NS21 | NS22 | NS23 | NS24 | NS25 | NS26 | NS27 | NS28 | NS29 | NS30 | NS31 | NS32 Value CA_NS_01 to CA_NS_32

get_value() → NetworkSigValueNoCarrAggr

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:NSValue
value: enums.NetworkSigValueNoCarrAggr = driver.configure.multiEval.nsValue.get_
↪value()
```

Selects the 'network signaled value' for measurements without carrier aggregation. For the combined signal path scenario, use CONFigure:LTE:SIGN<i>:CONNection:ASEMission.

return
value: NS01 | ... | NS288 Value NS_01 to NS_288

set_carrier_aggregation(*value: NetworkSigValue*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:NSValue:CAGGregation
driver.configure.multiEval.nsValue.set_carrier_aggregation(value = enums.
↪NetworkSigValue.NS01)
```

Selects the 'network signaled value' for measurements with carrier aggregation. For the combined signal path scenario, use CONFigure:LTE:SIGN<i>:CONNection:SCC<c>:ASEMission:CAGGregation.

param value
NS01 | NS02 | NS03 | NS04 | NS05 | NS06 | NS07 | NS08 | NS09 | NS10 | NS11 | NS12 | NS13 | NS14 | NS15 | NS16 | NS17 | NS18 | NS19 | NS20 | NS21 | NS22 | NS23 | NS24 | NS25 | NS26 | NS27 | NS28 | NS29 | NS30 | NS31 | NS32 Value CA_NS_01 to CA_NS_32

set_value(*value: NetworkSigValueNoCarrAggr*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:NSValue
driver.configure.multiEval.nsValue.set_value(value = enums.
↪NetworkSigValueNoCarrAggr.NS01)
```

Selects the 'network signaled value' for measurements without carrier aggregation. For the combined signal path scenario, use CONFigure:LTE:SIGN<i>:CONNection:ASEMission.

param value
NS01 | … | NS288 Value NS_01 to NS_288

6.1.4.8 Pcc

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation[:PCC]:PLCid
```

class PccCls

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

get_plc_id() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation[:PCC]:PLCid
value: int = driver.configure.multiEval.pcc.get_plc_id()
```

No command help available

return
phs_layer_cell_id: No help available

set_plc_id(*phs_layer_cell_id: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation[:PCC]:PLCid
driver.configure.multiEval.pcc.set_plc_id(phs_layer_cell_id = 1)
```

No command help available

param phs_layer_cell_id
No help available

6.1.4.9 Pdynamics

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:PDYNamics:TMASk
```

class PdynamicsCls

Pdynamics commands group definition. 3 total commands, 1 Subgroups, 1 group commands

get_tmask() → TimeMask

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:PDYNamics:TMASk
value: enums.TimeMask = driver.configure.multiEval.pdynamics.get_tmask()
```

Selects the time mask for power dynamics measurements.

return
time_mask: GOO | PPSRs | SBLanking GOO: General ON/OFF time mask PPSRs: PUCCH/PUSCH transmission before and after an SRS SBLanking: SRS blanking time mask

set_tmask(*time_mask: TimeMask*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:PDYNamics:TMASk
driver.configure.multiEval.pdynamics.set_tmask(time_mask = enums.TimeMask.GOO)
```

Selects the time mask for power dynamics measurements.

param time_mask
GOO | PPSRs | SBLanking GOO: General ON/OFF time mask PPSRs: PUCCH/PUSCH transmission before and after an SRS SBLanking: SRS blanking time mask

Cloning the Group

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

Subgroups

6.1.4.9.1 AeoPower

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:PDYNamics:AEOPower:LEADing
CONFigure:LTE:MEASurement<Instance>:MEValuation:PDYNamics:AEOPower:LAGGing
```

class AeoPowerCls

AeoPower commands group definition. 2 total commands, 0 Subgroups, 2 group commands

get_lagging() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:PDYNamics:AEOPower:LAGGing
value: int = driver.configure.multiEval.pdynamics.aeoPower.get_lagging()
```

Shifts the end of the evaluation period for OFF power measurements.

return
lagging: integer Positive values reduce the evaluation period (ends earlier) . Negative values increase the evaluation period (ends later) . Range: -1000 Ts to 1000 Ts, Unit: Ts

get_leading() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:PDYNamics:AEOPower:LEADing
value: int = driver.configure.multiEval.pdynamics.aeoPower.get_leading()
```

Shifts the beginning of the evaluation period for OFF power measurements.

return
leading: integer Positive values reduce the evaluation period (starts later) . Negative values increase the evaluation period (starts earlier) . Range: -1000 Ts to 1000 Ts, Unit: Ts

set_lagging(*lagging: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:PDYNamics:AEOPower:LAGGing
driver.configure.multiEval.pdynamics.aeoPower.set_lagging(lagging = 1)
```

Shifts the end of the evaluation period for OFF power measurements.

param lagging
integer Positive values reduce the evaluation period (ends earlier) . Negative values increase the evaluation period (ends later) . Range: -1000 Ts to 1000 Ts, Unit: Ts

set_leading(*leading: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:PDYNamics:AEOPower:LEADing
driver.configure.multiEval.pdynamics.aeoPower.set_leading(leading = 1)
```

Shifts the beginning of the evaluation period for OFF power measurements.

param leading
integer Positive values reduce the evaluation period (starts later) . Negative values increase the evaluation period (starts earlier) . Range: -1000 Ts to 1000 Ts, Unit: Ts

6.1.4.10 Power

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:POWer:HDMode
```

class PowerCls

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

get_hdmode() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:POWer:HDMode
value: bool = driver.configure.multiEval.power.get_hdmode()
```

Enables or disables the high dynamic mode for power dynamics measurements.

return
high_dynamic_mode: OFF | ON

set_hdmode(*high_dynamic_mode: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:POWer:HDMode
driver.configure.multiEval.power.set_hdmode(high_dynamic_mode = False)
```

Enables or disables the high dynamic mode for power dynamics measurements.

param high_dynamic_mode
OFF | ON

6.1.4.11 RbAllocation

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:AUTO
```

class RbAllocationCls

RbAllocation commands group definition. 10 total commands, 3 Subgroups, 1 group commands

get_auto() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:AUTO
value: bool = driver.configure.multiEval.rbAllocation.get_auto()
```

Enables or disables the automatic detection of the RB configuration.

return
auto: OFF | ON OFF: manual definition ON: automatic detection

set_auto(*auto: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:AUTO
driver.configure.multiEval.rbAllocation.set_auto(auto = False)
```

Enables or disables the automatic detection of the RB configuration.

param auto
OFF | ON OFF: manual definition ON: automatic detection

Cloning the Group

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

Subgroups

6.1.4.11.1 Mcluster

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:MCLuster
```

class MclusterCls

Mcluster commands group definition. 3 total commands, 2 Subgroups, 1 group commands

get_value() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:MCLuster
value: bool = driver.configure.multiEval.rbAllocation.mcluster.get_value()
```

Specifies whether the UL signal uses multi-cluster allocation or not.
INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:LTE:SIGN<i>:CONNection[:PCC]:MCLuster:UL
- CONFigure:LTE:SIGN<i>:CONNection:SCC<c>:MCLuster:UL

return
enable: OFF | ON OFF: contiguous allocation, resource allocation type 0 ON: multi-cluster allocation, resource allocation type 1

set_value(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:MCLuster
driver.configure.multiEval.rbAllocation.mcluster.set_value(enable = False)
```

Specifies whether the UL signal uses multi-cluster allocation or not.
INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:LTE:SIGN<i>:CONNection[:PCC]:MCLuster:UL
- CONFigure:LTE:SIGN<i>:CONNection:SCC<c>:MCLuster:UL

param enable
OFF | ON OFF: contiguous allocation, resource allocation type 0 ON: multi-cluster allocation, resource allocation type 1

Cloning the Group

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

Subgroups

6.1.4.11.1.1 Nrb<RBcount>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.configure.multiEval.rbAllocation.mcluster.nrb.repcap_rBcount_get()
driver.configure.multiEval.rbAllocation.mcluster.nrb.repcap_rBcount_set(repcap.RBcount.
↪Nr1)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:MCLuster:NRB<Number>
```

class NrbCls

Nrb commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: RBcount, default value after init: RBcount.Nr1

get(*rBcount=RBcount.Default*) → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:RBALlocation:MCLuster:NRB<Number>
value: int = driver.configure.multiEval.rbAllocation.mcluster.nrb.get(rBcount =␣
↪repcap.RBcount.Default)
```

Specifies the number of allocated RBs in the measured slot, for multi-cluster allocation.

INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:LTE:SIGN<i>:CONNection[:PCC]:RMC:MCLuster:UL
- CONFigure:LTE:SIGN<i>:CONNection[:PCC]:UDCHannels:MCLuster:UL
- CONFigure:LTE:SIGN<i>:CONNection:SCC<c>:RMC:MCLuster:UL
- CONFigure:LTE:SIGN<i>:CONNection:SCC<c>:UDCHannels:MCLuster:UL

param rBcount
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Nrb')

return
no_rb: numeric For the allowed input ranges, see 'Uplink resource block allocation'.

set(*no_rb: int*, *rBcount=RBcount.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:RBALlocation:MCLuster:NRB<Number>
driver.configure.multiEval.rbAllocation.mcluster.nrb.set(no_rb = 1, rBcount =␣
↪repcap.RBcount.Default)
```

Specifies the number of allocated RBs in the measured slot, for multi-cluster allocation.

INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:LTE:SIGN<i>:CONNection[:PCC]:RMC:MCLuster:UL
- CONFigure:LTE:SIGN<i>:CONNection[:PCC]:UDCHannels:MCLuster:UL
- CONFigure:LTE:SIGN<i>:CONNection:SCC<c>:RMC:MCLuster:UL
- CONFigure:LTE:SIGN<i>:CONNection:SCC<c>:UDCHannels:MCLuster:UL

param no_rb
numeric For the allowed input ranges, see 'Uplink resource block allocation'.

param rBcount
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Nrb')

Cloning the Group

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

6.1.4.11.1.2 Orb<RBoffset>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.configure.multiEval.rbAllocation.mcluster.orb.repcap_rBoffset_get()
driver.configure.multiEval.rbAllocation.mcluster.orb.repcap_rBoffset_set(repcap.RBoffset.
↪Nr1)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:MCLuster:ORB<Number>
```

class OrbCls

Orb commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: RBoffset, default value after init: RBoffset.Nr1

get(*rBoffset=RBoffset.Default*) → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:RBALlocation:MCLuster:ORB<Number>
value: int = driver.configure.multiEval.rbAllocation.mcluster.orb.get(rBoffset␣
↪= repcap.RBoffset.Default)
```

Specifies the offset of the first allocated resource block, for multi-cluster allocation.

INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:LTE:SIGN<i>:CONNection[:PCC]:RMC:MCLuster:UL
- CONFigure:LTE:SIGN<i>:CONNection[:PCC]:UDCHannels:MCLuster:UL
- CONFigure:LTE:SIGN<i>:CONNection:SCC<c>:RMC:MCLuster:UL
- CONFigure:LTE:SIGN<i>:CONNection:SCC<c>:UDCHannels:MCLuster:UL

param rBoffset
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Orb')

return
offset_rb: numeric For the allowed input ranges, see 'Uplink resource block allocation'.

set(*offset_rb: int*, *rBoffset=RBoffset.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:RBALlocation:MCLuster:ORB<Number>
driver.configure.multiEval.rbAllocation.mcluster.orb.set(offset_rb = 1,␣
↪rBoffset = repcap.RBoffset.Default)
```

Specifies the offset of the first allocated resource block, for multi-cluster allocation.
INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:LTE:SIGN<i>:CONNection[:PCC]:RMC:MCLuster:UL
- CONFigure:LTE:SIGN<i>:CONNection[:PCC]:UDCHannels:MCLuster:UL
- CONFigure:LTE:SIGN<i>:CONNection:SCC<c>:RMC:MCLuster:UL
- CONFigure:LTE:SIGN<i>:CONNection:SCC<c>:UDCHannels:MCLuster:UL

param offset_rb
numeric For the allowed input ranges, see 'Uplink resource block allocation'.

param rBoffset
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Orb')

Cloning the Group

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

6.1.4.11.2 Nrb

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:NRB:PSCCh
CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:NRB:PSSCh
CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:NRB
```

class NrbCls

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

get_pscch() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:NRB:PSCCh
value: int = driver.configure.multiEval.rbAllocation.nrb.get_pscch()
```

Specifies the number of allocated RBs for the PSCCH in the measured slot. For manual RB allocation definition, for sidelink signals.

return
no_rb: numeric The value is fixed. Range: 2 to 2

get_pssch() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:NRB:PSSCh
value: int = driver.configure.multiEval.rbAllocation.nrb.get_pssch()
```

Specifies the number of allocated RBs for the PSSCH in the measured slot. For manual RB allocation definition, for sidelink signals.

return
no_rb: numeric For the allowed input range, see 'Sidelink resource block allocation'.

get_value() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:NRB
value: int = driver.configure.multiEval.rbAllocation.nrb.get_value()
```

Specifies the number of allocated RBs in the measured slot. For manual RB allocation definition, for uplink signals without multi-cluster allocation.

return
no_rb: numeric For the allowed input range, see 'Uplink resource block allocation'.

set_pscch(*no_rb: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:NRB:PSCCh
driver.configure.multiEval.rbAllocation.nrb.set_pscch(no_rb = 1)
```

Specifies the number of allocated RBs for the PSCCH in the measured slot. For manual RB allocation definition, for sidelink signals.

param no_rb
numeric The value is fixed. Range: 2 to 2

set_pssch(*no_rb: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:NRB:PSSCh
driver.configure.multiEval.rbAllocation.nrb.set_pssch(no_rb = 1)
```

Specifies the number of allocated RBs for the PSSCH in the measured slot. For manual RB allocation definition, for sidelink signals.

param no_rb
numeric For the allowed input range, see 'Sidelink resource block allocation'.

set_value(*no_rb: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:NRB
driver.configure.multiEval.rbAllocation.nrb.set_value(no_rb = 1)
```

Specifies the number of allocated RBs in the measured slot. For manual RB allocation definition, for uplink signals without multi-cluster allocation.

param no_rb
numeric For the allowed input range, see 'Uplink resource block allocation'.

6.1.4.11.3 Orb

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:ORB:PSCCh
CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:ORB:PSSCh
CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:ORB
```

class OrbCls

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

get_pscch() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:ORB:PSCCh
value: int = driver.configure.multiEval.rbAllocation.orb.get_pscch()
```

Specifies the offset of the first allocated PSCCH resource block for manual RB allocation definition, for sidelink signals.

return
offset_rb: numeric For the maximum number of RBs depending on the channel BW, see 'Uplink resource block allocation'. Range: 0 to max no of RBs minus total allocated RBs

get_pssch() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:ORB:PSSCh
value: int = driver.configure.multiEval.rbAllocation.orb.get_pssch()
```

Specifies the offset of the first allocated PSSCH resource block for manual RB allocation definition, for sidelink signals.

return
offset_rb: numeric The range depends on the OffsetRB for the PSCCH, the channel BW and the number of allocated PSSCH RBs, see 'Sidelink resource block allocation'.

get_value() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:ORB
value: int = driver.configure.multiEval.rbAllocation.orb.get_value()
```

Specifies the offset of the first allocated resource block for manual RB allocation definition, for uplink signals without multi-cluster allocation.

return
offset_rb: numeric For the maximum number of RBs depending on the channel BW, see 'Uplink resource block allocation'. Range: 0 to maximum number of RBs minus 1

set_pscch(*offset_rb: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:ORB:PSCCh
driver.configure.multiEval.rbAllocation.orb.set_pscch(offset_rb = 1)
```

Specifies the offset of the first allocated PSCCH resource block for manual RB allocation definition, for sidelink signals.

param offset_rb
numeric For the maximum number of RBs depending on the channel BW, see 'Uplink resource block allocation'. Range: 0 to max no of RBs minus total allocated RBs

set_pssch(*offset_rb: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:ORB:PSSCh
driver.configure.multiEval.rbAllocation.orb.set_pssch(offset_rb = 1)
```

Specifies the offset of the first allocated PSSCH resource block for manual RB allocation definition, for sidelink signals.

param offset_rb
numeric The range depends on the OffsetRB for the PSCCH, the channel BW and the number of allocated PSSCH RBs, see 'Sidelink resource block allocation'.

`set_value`(*offset_rb: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:ORB
driver.configure.multiEval.rbAllocation.orb.set_value(offset_rb = 1)
```

Specifies the offset of the first allocated resource block for manual RB allocation definition, for uplink signals without multi-cluster allocation.

param offset_rb
numeric For the maximum number of RBs depending on the channel BW, see 'Uplink resource block allocation'. Range: 0 to maximum number of RBs minus 1

6.1.4.12 Result

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult[:ALL]
CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:MERRor
CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:PERRor
CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:IEMissions
CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:EVMC
CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:ESFLatness
CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:TXM
CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:IQ
CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:SEMask
CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:ACLR
CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:RBATable
CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:PMONitor
CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:PDYNamics
CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:BLER
```

`class ResultCls`

Result commands group definition. 16 total commands, 1 Subgroups, 14 group commands

`class AllStruct`

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

- Evm: bool: OFF | ON Error vector magnitude OFF: Do not evaluate results. ON: Evaluate results.
- Magnitude_Error: bool: OFF | ON
- Phase_Error: bool: OFF | ON
- Inband_Emissions: bool: OFF | ON
- Evm_Versus_C: bool: OFF | ON EVM vs subcarrier
- Iq: bool: OFF | ON I/Q constellation diagram
- Equ_Spec_Flatness: bool: OFF | ON Equalizer spectrum flatness
- Tx_Measurement: bool: OFF | ON TX measurement statistical overview
- Spec_Em_Mask: bool: OFF | ON Spectrum emission mask

- Aclr: bool: OFF | ON Adjacent channel leakage power ratio
- Rb_Alloc_Table: bool: Optional setting parameter. OFF | ON Resource block allocation table
- Power_Monitor: bool: Optional setting parameter. OFF | ON
- Bler: bool: Optional setting parameter. OFF | ON Block error ratio
- Power_Dynamics: bool: Optional setting parameter. OFF | ON

get_aclr() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:ACLR
value: bool = driver.configure.multiEval.result.get_aclr()
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / Inband emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
- BLER / Block error ratio
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCmwLteMeas.Configure.MultiEval.Result.all.

return
enable: OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

get_all() → AllStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult[:ALL]
value: AllStruct = driver.configure.multiEval.result.get_all()
```

Enables or disables the evaluation of results in the multi-evaluation measurement. This command combines most other CONFigure:LTE:MEAS<i>:MEValuation:RESult... commands.

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

get_bler() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:BLER
value: bool = driver.configure.multiEval.result.get_bler()
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / Inband emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
- BLER / Block error ratio
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCmwLteMeas.Configure.MultiEval.Result.all.

return
enable: OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

get_es_flatness() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:ESFLatness
value: bool = driver.configure.multiEval.result.get_es_flatness()
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / Inband emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
- BLER / Block error ratio
- EVMC / EVM vs subcarrier

- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCmwLteMeas.Configure.MultiEval.Result.all.

return
enable: OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

get_evmc() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:EVMC
value: bool = driver.configure.multiEval.result.get_evmc()
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / Inband emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
- BLER / Block error ratio
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCmwLteMeas.Configure.MultiEval.Result.all.

return
enable: OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

get_iemissions() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:IEMissions
value: bool = driver.configure.multiEval.result.get_iemissions()
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / Inband emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
- BLER / Block error ratio
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCmwLteMeas.Configure.MultiEval.Result.all.

return
enable: OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

`get_iq`() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:IQ
value: bool = driver.configure.multiEval.result.get_iq()
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / Inband emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
- BLER / Block error ratio
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio

- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCmwLteMeas.Configure.MultiEval.Result.all.

return
enable: OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

get_merror() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:MERRor
value: bool = driver.configure.multiEval.result.get_merror()
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / Inband emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
- BLER / Block error ratio
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCmwLteMeas.Configure.MultiEval.Result.all.

return
enable: OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

get_pdynamics() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:PDYNamics
value: bool = driver.configure.multiEval.result.get_pdynamics()
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / Inband emissions

- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
- BLER / Block error ratio
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCmwLteMeas.Configure.MultiEval.Result.all.

return
enable: OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

get_perror() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:PERRor
value: bool = driver.configure.multiEval.result.get_perror()
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / Inband emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
- BLER / Block error ratio
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCmwLteMeas.Configure.MultiEval.Result.all.

return
enable: OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

get_pmonitor() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:PMONitor
value: bool = driver.configure.multiEval.result.get_pmonitor()
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / Inband emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
- BLER / Block error ratio
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCmwLteMeas.Configure.MultiEval.Result.all.

return
enable: OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

get_rba_table() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:RBATable
value: bool = driver.configure.multiEval.result.get_rba_table()
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / Inband emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
- BLER / Block error ratio
- EVMC / EVM vs subcarrier

- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCmwLteMeas.Configure.MultiEval.Result.all.

return
enable: OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

get_se_mask() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:SEMask
value: bool = driver.configure.multiEval.result.get_se_mask()
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / Inband emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
- BLER / Block error ratio
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCmwLteMeas.Configure.MultiEval.Result.all.

return
enable: OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

get_txm() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:TXM
value: bool = driver.configure.multiEval.result.get_txm()
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / Inband emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
- BLER / Block error ratio
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCmwLteMeas.Configure.MultiEval.Result.all.

return
enable: OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

set_aclr(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:ACLR
driver.configure.multiEval.result.set_aclr(enable = False)
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / Inband emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
- BLER / Block error ratio
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio

- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCmwLteMeas.Configure.MultiEval.Result.all.

param enable
OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

set_all(*value: AllStruct*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult[:ALL]
structure = driver.configure.multiEval.result.AllStruct()
structure.Evm: bool = False
structure.Magnitude_Error: bool = False
structure.Phase_Error: bool = False
structure.Inband_Emissions: bool = False
structure.Evm_Versus_C: bool = False
structure.Iq: bool = False
structure.Equ_Spec_Flatness: bool = False
structure.Tx_Measurement: bool = False
structure.Spec_Em_Mask: bool = False
structure.Aclr: bool = False
structure.Rb_Alloc_Table: bool = False
structure.Power_Monitor: bool = False
structure.Bler: bool = False
structure.Power_Dynamics: bool = False
driver.configure.multiEval.result.set_all(value = structure)
```

Enables or disables the evaluation of results in the multi-evaluation measurement. This command combines most other CONFigure:LTE:MEAS<i>:MEValuation:RESult... commands.

param value
see the help for AllStruct structure arguments.

set_bler(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:BLER
driver.configure.multiEval.result.set_bler(enable = False)
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / Inband emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
- BLER / Block error ratio
- EVMC / EVM vs subcarrier
- PERRor / Phase error

- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCmwLteMeas.Configure.MultiEval.Result.all.

param enable
OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

set_es_flatness(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:ESFLatness
driver.configure.multiEval.result.set_es_flatness(enable = False)
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / Inband emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
- BLER / Block error ratio
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCmwLteMeas.Configure.MultiEval.Result.all.

param enable
OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

set_evmc(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:EVMC
driver.configure.multiEval.result.set_evmc(enable = False)
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / Inband emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
- BLER / Block error ratio
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCmwLteMeas.Configure.MultiEval.Result.all.

param enable
OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

set_iemissions(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:IEMissions
driver.configure.multiEval.result.set_iemissions(enable = False)
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / Inband emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
- BLER / Block error ratio
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCmwLteMeas.Configure.MultiEval.Result.all.

param enable
OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

set_iq(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:IQ
driver.configure.multiEval.result.set_iq(enable = False)
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / Inband emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
- BLER / Block error ratio
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCmwLteMeas.Configure.MultiEval.Result.all.

param enable
OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

set_merror(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:MERRor
driver.configure.multiEval.result.set_merror(enable = False)
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / Inband emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table

- BLER / Block error ratio
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCmwLteMeas.Configure.MultiEval.Result.all.

param enable
OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

set_pdynamics(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:PDYNamics
driver.configure.multiEval.result.set_pdynamics(enable = False)
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / Inband emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
- BLER / Block error ratio
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCmwLteMeas.Configure.MultiEval.Result.all.

param enable
OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

set_perror(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:PERRor
driver.configure.multiEval.result.set_perror(enable = False)
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / Inband emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
- BLER / Block error ratio
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCmwLteMeas.Configure.MultiEval.Result.all.

param enable
OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

set_pmonitor(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:PMONitor
driver.configure.multiEval.result.set_pmonitor(enable = False)
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / Inband emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
- BLER / Block error ratio
- EVMC / EVM vs subcarrier

- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCmwLteMeas.Configure.MultiEval.Result.all.

param enable
OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

set_rba_table(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:RBATable
driver.configure.multiEval.result.set_rba_table(enable = False)
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / Inband emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
- BLER / Block error ratio
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCmwLteMeas.Configure.MultiEval.Result.all.

param enable
OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

set_se_mask(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:SEMask
driver.configure.multiEval.result.set_se_mask(enable = False)
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / Inband emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
- BLER / Block error ratio
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCmwLteMeas.Configure.MultiEval.Result.all.

param enable
OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

set_txm(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:TXM
driver.configure.multiEval.result.set_txm(enable = False)
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / Inband emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
- BLER / Block error ratio
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio

- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCmwLteMeas.Configure.MultiEval.Result.all.

param enable
OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

Cloning the Group

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

Subgroups

6.1.4.12.1 EvMagnitude

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:EVMagnitude
```

class EvMagnitudeCls

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

get_value() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:EVMagnitude
value: bool = driver.configure.multiEval.result.evMagnitude.get_value()
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / Inband emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
- BLER / Block error ratio
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor

- PDYNamics / Power dynamics

For reset values, see method RsCmwLteMeas.Configure.MultiEval.Result.all.

return
enable: OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

`set_value`(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:EVMagnitude
driver.configure.multiEval.result.evMagnitude.set_value(enable = False)
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / Inband emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
- BLER / Block error ratio
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCmwLteMeas.Configure.MultiEval.Result.all.

param enable
OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

Cloning the Group

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

Subgroups

6.1.4.12.1.1 EvmSymbol

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:EVMagnitude:EVMSymbol
```

class EvmSymbolCls

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

class EvmSymbolStruct

Response structure. Fields:

- Enable: bool: OFF | ON OFF: Do not measure the results. ON: Measure the results.
- Symbol: int: decimal SC-FDMA symbol to be evaluated Range: 0 to 6
- Low_High: enums.LowHigh: LOW | HIGH Low or high EVM window position

get() → EvmSymbolStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:RESult:EVMagnitude:EVMSymbol
value: EvmSymbolStruct = driver.configure.multiEval.result.evMagnitude.
↪evmSymbol.get()
```

Enables or disables the measurement of EVM vs modulation symbol results and configures the scope of the measurement.

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

set(*enable: bool*, *symbol: int*, *low_high: LowHigh*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:RESult:EVMagnitude:EVMSymbol
driver.configure.multiEval.result.evMagnitude.evmSymbol.set(enable = False,␣
↪symbol = 1, low_high = enums.LowHigh.HIGH)
```

Enables or disables the measurement of EVM vs modulation symbol results and configures the scope of the measurement.

param enable
OFF | ON OFF: Do not measure the results. ON: Measure the results.

param symbol
decimal SC-FDMA symbol to be evaluated Range: 0 to 6

param low_high
LOW | HIGH Low or high EVM window position

6.1.4.13 Scount

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:SCOunt:MODulation
CONFigure:LTE:MEASurement<Instance>:MEValuation:SCOunt:POWer
```

class ScountCls

Scount commands group definition. 4 total commands, 1 Subgroups, 2 group commands

get_modulation() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:SCOunt:MODulation
value: int = driver.configure.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: numeric Range: 1 slot to 1000 slots

get_power() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:SCOunt:POWer
value: int = driver.configure.multiEval.scount.get_power()
```

Specifies the statistic count of the measurement. The statistic count is equal to the number of measurement intervals per single shot.

return
statistic_count: numeric Range: 1 subframe to 1000 subframes

set_modulation(*statistic_count: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:SCOunt:MODulation
driver.configure.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
numeric Range: 1 slot to 1000 slots

set_power(*statistic_count: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:SCOunt:POWer
driver.configure.multiEval.scount.set_power(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
numeric Range: 1 subframe to 1000 subframes

Cloning the Group

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

Subgroups

6.1.4.13.1 Spectrum

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:SCOunt:SPECtrum:SEMask
CONFigure:LTE:MEASurement<Instance>:MEValuation:SCOunt:SPECtrum:ACLR
```

class SpectrumCls

Spectrum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

get_aclr() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:SCOunt:SPECtrum:ACLR
value: int = driver.configure.multiEval.scount.spectrum.get_aclr()
```

Specifies the statistic count of the measurement. The statistic count is equal to the number of measurement intervals per single shot. Separate statistic counts for ACLR and spectrum emission mask measurements are supported.

return
statistic_count: numeric Range: 1 slot to 1000 slots

get_se_mask() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:SCOunt:SPECtrum:SEMask
value: int = driver.configure.multiEval.scount.spectrum.get_se_mask()
```

Specifies the statistic count of the measurement. The statistic count is equal to the number of measurement intervals per single shot. Separate statistic counts for ACLR and spectrum emission mask measurements are supported.

return
statistic_count: numeric Range: 1 slot to 1000 slots

set_aclr(*statistic_count: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:SCOunt:SPECtrum:ACLR
driver.configure.multiEval.scount.spectrum.set_aclr(statistic_count = 1)
```

Specifies the statistic count of the measurement. The statistic count is equal to the number of measurement intervals per single shot. Separate statistic counts for ACLR and spectrum emission mask measurements are supported.

param statistic_count
numeric Range: 1 slot to 1000 slots

set_se_mask(*statistic_count: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:SCOunt:SPECtrum:SEMask
driver.configure.multiEval.scount.spectrum.set_se_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. Separate statistic counts for ACLR and spectrum emission mask measurements are supported.

param statistic_count
numeric Range: 1 slot to 1000 slots

6.1.4.14 Spectrum

class SpectrumCls

Spectrum 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.configure.multiEval.spectrum.clone()
```

Subgroups

6.1.4.14.1 Aclr

class AclrCls

Aclr 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.multiEval.spectrum.aclr.clone()
```

Subgroups

6.1.4.14.1.1 Enable

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:SPECtrum:ACLR:ENABle
```

class EnableCls

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

class EnableStruct

Response structure. Fields:

- Utra_1: bool: OFF | ON
- Utra_2: bool: OFF | ON
- Eutra: bool: OFF | ON

get() → EnableStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:SPECtrum:ACLR:ENABle
value: EnableStruct = driver.configure.multiEval.spectrum.aclr.enable.get()
```

Enables or disables the evaluation of the first adjacent UTRA channels, second adjacent UTRA channels and first adjacent E-UTRA channels.

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

set(*utra_1: bool*, *utra_2: bool*, *eutra: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:SPECtrum:ACLR:ENABle
driver.configure.multiEval.spectrum.aclr.enable.set(utra_1 = False, utra_2 =␣
↪False, eutra = False)
```

Enables or disables the evaluation of the first adjacent UTRA channels, second adjacent UTRA channels and first adjacent E-UTRA channels.

param utra_1
OFF | ON

param utra_2
OFF | ON

param eutra
OFF | ON

6.1.4.14.2 SeMask

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:SPECtrum:SEMask:MFILter
```

class SeMaskCls

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

get_mfilter() → MeasFilter

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:SPECtrum:SEMask:MFILter
value: enums.MeasFilter = driver.configure.multiEval.spectrum.seMask.get_
↪mfilter()
```

Selects the resolution filter type for filter bandwidths of 50 kHz and greater.

return
meas_filter: BANDpass | GAUSs

set_mfilter(*meas_filter: MeasFilter*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:SPECtrum:SEMask:MFILter
driver.configure.multiEval.spectrum.seMask.set_mfilter(meas_filter = enums.
↪MeasFilter.BANDpass)
```

Selects the resolution filter type for filter bandwidths of 50 kHz and greater.

param meas_filter
BANDpass | GAUSs

6.1.4.15 Srs

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:SRS:ENABle
```

class SrsCls

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

get_enable() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:SRS:ENABle
value: bool = driver.configure.multiEval.srs.get_enable()
```

Specifies whether a sounding reference signal is allowed (ON) or not (OFF) . For the combined signal path scenario, use CONFigure:LTE:SIGN<i>:CELL[:PCC]:SRS:ENABle.

return
enable: OFF | ON OFF: no SRS signal ON: SRS signal allowed in the last SC-FDMA symbol of each subframe

set_enable(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:SRS:ENABle
driver.configure.multiEval.srs.set_enable(enable = False)
```

Specifies whether a sounding reference signal is allowed (ON) or not (OFF) . For the combined signal path scenario, use CONFigure:LTE:SIGN<i>:CELL[:PCC]:SRS:ENABle.

param enable
OFF | ON OFF: no SRS signal ON: SRS signal allowed in the last SC-FDMA symbol of each subframe

6.1.4.16 Tmode

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:TMODe:SCOunt
CONFigure:LTE:MEASurement<Instance>:MEValuation:TMODe:ENPower
CONFigure:LTE:MEASurement<Instance>:MEValuation:TMODe:RLEVel
```

class TmodeCls

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

get_envelope_power() → List[float]

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:TMODe:ENPower
value: List[float] = driver.configure.multiEval.tmode.get_envelope_power()
```

Defines the expected nominal power values for all entries of the 'TPC Mode' list. For definition of the corresponding subframe count values, see method RsCmwLteMeas.Configure.MultiEval.Tmode.scount.

return
exp_nom_pow: numeric Comma-separated list of 16 values, for list entry number 0 to 15 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 data sheet. Unit: dBm

get_rlevel() → List[float]

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:TMODe:RLEVel
value: List[float] = driver.configure.multiEval.tmode.get_rlevel()
```

Queries the reference level for all entries of the 'TPC Mode' list. The reference level is calculated from the expected nominal power of each entry and the user margin.

return
reference_level: float Comma-separated list of 16 values, for list entry number 0 to 15 The range of the reference levels can be calculated as follows: Range (Reference Level) = Range (Input Power) + External Attenuation The input power range is stated in the data sheet. Unit: dBm

get_scount() → List[int]

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:TMODe:SCOunt
value: List[int] = driver.configure.multiEval.tmode.get_scount()
```

Defines the subframe counts for all entries of the 'TPC Mode' list. For definition of the corresponding expected nominal power values, see method RsCmwLteMeas.Configure.MultiEval.Tmode.envelopePower.

return
sub_frame_count: decimal Comma-separated list of 16 values, for list entry number 0 to 15 Range: 1 to 320

set_envelope_power(*exp_nom_pow: List[float]*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:TMODe:ENPower
driver.configure.multiEval.tmode.set_envelope_power(exp_nom_pow = [1.1, 2.2, 3.
↪3])
```

Defines the expected nominal power values for all entries of the 'TPC Mode' list. For definition of the corresponding subframe count values, see method RsCmwLteMeas.Configure.MultiEval.Tmode.scount.

param exp_nom_pow
numeric Comma-separated list of 16 values, for list entry number 0 to 15 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 data sheet. Unit: dBm

set_scount(*sub_frame_count: List[int]*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:TMODe:SCOunt
driver.configure.multiEval.tmode.set_scount(sub_frame_count = [1, 2, 3])
```

Defines the subframe counts for all entries of the 'TPC Mode' list. For definition of the corresponding expected nominal power values, see method RsCmwLteMeas.Configure.MultiEval.Tmode.envelopePower.

param sub_frame_count
decimal Comma-separated list of 16 values, for list entry number 0 to 15 Range: 1 to 320

6.1.5 Network

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:NETWork:RFPSharing
CONFigure:LTE:MEASurement<Instance>:NETWork:DMODe
```

class NetworkCls

Network commands group definition. 2 total commands, 0 Subgroups, 2 group commands

get_dmode() → DuplexMode

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:NETWork:DMODe
value: enums.DuplexMode = driver.configure.network.get_dmode()
```

No command help available

return
mode: No help available

get_rfp_sharing() → Sharing

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:NETWork:RFPSharing
value: enums.Sharing = driver.configure.network.get_rfp_sharing()
```

No command help available

return
sharing: No help available

set_dmode(*mode: DuplexMode*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:NETWork:DMODe
driver.configure.network.set_dmode(mode = enums.DuplexMode.FDD)
```

No command help available

param mode
No help available

set_rfp_sharing(*sharing: Sharing*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:NETWork:RFPSharing
driver.configure.network.set_rfp_sharing(sharing = enums.Sharing.FSHared)
```

No command help available

param sharing
No help available

6.1.6 Pcc

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>[:PCC]:CBANdwidth
```

class PccCls

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

get_channel_bw() → ChannelBandwidth

```
# SCPI: CONFigure:LTE:MEASurement<Instance>[:PCC]:CBANdwidth
value: enums.ChannelBandwidth = driver.configure.pcc.get_channel_bw()
```

No command help available

return
channel_bw: No help available

set_channel_bw(*channel_bw: ChannelBandwidth*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>[:PCC]:CBANdwidth
driver.configure.pcc.set_channel_bw(channel_bw = enums.ChannelBandwidth.B014)
```

No command help available

param channel_bw
No help available

6.1.7 Prach

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:PRACh:TOUT
CONFigure:LTE:MEASurement<Instance>:PRACh:REPetition
CONFigure:LTE:MEASurement<Instance>:PRACh:SCONdition
CONFigure:LTE:MEASurement<Instance>:PRACh:MOEXception
CONFigure:LTE:MEASurement<Instance>:PRACh:PCINdex
CONFigure:LTE:MEASurement<Instance>:PRACh:SSYMbol
CONFigure:LTE:MEASurement<Instance>:PRACh:NOPReambles
CONFigure:LTE:MEASurement<Instance>:PRACh:POPReambles
```

class PrachCls

Prach commands group definition. 34 total commands, 6 Subgroups, 8 group commands

get_mo_exception() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:MOEXception
value: bool = driver.configure.prach.get_mo_exception()
```

Specifies whether measurement results that the R&S CMW identifies as faulty or inaccurate are rejected.

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

get_no_preambles() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:NOPReambles
value: int = driver.configure.prach.get_no_preambles()
```

Specifies the number of preambles to be captured per measurement interval.

return
number_preamble: numeric Range: 1 to 400

get_pc_index() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:PCINdex
value: int = driver.configure.prach.get_pc_index()
```

The PRACH configuration index identifies the PRACH configuration used by the UE (preamble format, which resources in the time domain are allowed for transmission of preambles etc.) .

INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:LTE:SIGN<i>:CELL:PRACh:PCINdex:FDD
- CONFigure:LTE:SIGN<i>:CELL:PRACh:PCINdex:TDD

return
prach_conf_index: numeric Range: 0 to 63 for FDD / 57 for TDD

get_po_preambles() → PeriodPreamble

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:POPReambles
value: enums.PeriodPreamble = driver.configure.prach.get_po_preambles()
```

Specifies the periodicity of preambles to be captured for multi-preamble result views.

return
period_preamble: MS05 | MS10 | MS20 MS05: 5 ms MS10: 10 ms MS20: 20 ms

get_repetition() → Repeat

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:REPetition
value: enums.Repeat = driver.configure.prach.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 | CONTinuous SINGleshot: Single-shot measurement CONTinuous: Continuous measurement

get_scondition() → StopCondition

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:SCONdition
value: enums.StopCondition = driver.configure.prach.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 | SLFail NONE: Continue measurement irrespective of the limit check. SLFail: Stop measurement on limit failure.

get_ssymbol() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:SSYMbol
value: int = driver.configure.prach.get_ssymbol()
```

Selects the OFDM symbol to be evaluated for single-symbol modulation result diagrams. The number of OFDM symbols in the preamble (<no of symbols>) depends on the preamble format, see Table 'Preambles in the time domain'.

return
selected_symbol: numeric Range: 0 to no of symbols -1

get_timeout() → float

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:TOUT
value: float = driver.configure.prach.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.

return
timeout: numeric Unit: s

set_mo_exception(*meas_on_exception: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:MOEXception
driver.configure.prach.set_mo_exception(meas_on_exception = False)
```

Specifies whether measurement results that the R&S CMW identifies as faulty or inaccurate are rejected.

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

set_no_preambles(*number_preamble: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:NOPReambles
driver.configure.prach.set_no_preambles(number_preamble = 1)
```

Specifies the number of preambles to be captured per measurement interval.

param number_preamble
numeric Range: 1 to 400

set_pc_index(*prach_conf_index: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:PCINdex
driver.configure.prach.set_pc_index(prach_conf_index = 1)
```

The PRACH configuration index identifies the PRACH configuration used by the UE (preamble format, which resources in the time domain are allowed for transmission of preambles etc.) .

INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:LTE:SIGN<i>:CELL:PRACh:PCINdex:FDD
- CONFigure:LTE:SIGN<i>:CELL:PRACh:PCINdex:TDD

param prach_conf_index
numeric Range: 0 to 63 for FDD / 57 for TDD

set_po_preambles(*period_preamble: PeriodPreamble*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:POPReambles
driver.configure.prach.set_po_preambles(period_preamble = enums.PeriodPreamble.
↪MS05)
```

Specifies the periodicity of preambles to be captured for multi-preamble result views.

param period_preamble
MS05 | MS10 | MS20 MS05: 5 ms MS10: 10 ms MS20: 20 ms

set_repetition(*repetition: Repeat*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:REPetition
driver.configure.prach.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 | CONTinuous SINGleshot: Single-shot measurement CONTinuous: Continuous measurement

set_scondition(*stop_condition: StopCondition*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:SCONdition
driver.configure.prach.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 | SLFail NONE: Continue measurement irrespective of the limit check. SLFail: Stop measurement on limit failure.

set_ssymbol(*selected_symbol: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:SSYMbol
driver.configure.prach.set_ssymbol(selected_symbol = 1)
```

Selects the OFDM symbol to be evaluated for single-symbol modulation result diagrams. The number of OFDM symbols in the preamble (<no of symbols>) depends on the preamble format, see Table 'Preambles in the time domain'.

param selected_symbol
numeric Range: 0 to no of symbols -1

set_timeout(*timeout: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:TOUT
driver.configure.prach.set_timeout(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.

param timeout
numeric Unit: s

Cloning the Group

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

Subgroups

6.1.7.1 Limit

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:PRACh:LIMit:FERRor
```

class LimitCls

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

get_freq_error() → float

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:LIMit:FERRor
value: float or bool = driver.configure.prach.limit.get_freq_error()
```

Defines an upper limit for the carrier frequency error.

return
frequency_error: (float or boolean) numeric | ON | OFF Range: 0 ppm to 1000 ppm, Unit: ppm ON | OFF enables or disables the limit check.

set_freq_error(*frequency_error: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:LIMit:FERRor
driver.configure.prach.limit.set_freq_error(frequency_error = 1.0)
```

Defines an upper limit for the carrier frequency error.

param frequency_error
(float or boolean) numeric | ON | OFF Range: 0 ppm to 1000 ppm, Unit: ppm ON | OFF enables or disables the limit check.

Cloning the Group

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

Subgroups

6.1.7.1.1 EvMagnitude

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:PRACh:LIMit:EVMagnitude
```

class EvMagnitudeCls

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

class EvMagnitudeStruct

Response structure. Fields:

- Rms: float or bool: numeric | ON | OFF Range: 0 % to 100 %, Unit: % ON | OFF enables or disables the limit check.
- Peak: float or bool: numeric | ON | OFF Range: 0 % to 100 %, Unit: % ON | OFF enables or disables the limit check.

get() → EvMagnitudeStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:LIMit:EVMagnitude
value: EvMagnitudeStruct = driver.configure.prach.limit.evMagnitude.get()
```

Defines upper limits for the RMS and peak values of the error vector magnitude (EVM) .

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

set(*rms: float*, *peak: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:LIMit:EVMagnitude
driver.configure.prach.limit.evMagnitude.set(rms = 1.0, peak = 1.0)
```

Defines upper limits for the RMS and peak values of the error vector magnitude (EVM) .

param rms
(float or boolean) numeric | ON | OFF Range: 0 % to 100 %, Unit: % ON | OFF enables or disables the limit check.

param peak
(float or boolean) numeric | ON | OFF Range: 0 % to 100 %, Unit: % ON | OFF enables or disables the limit check.

6.1.7.1.2 Merror

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:PRACh:LIMit:MERRor
```

class MerrorCls

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

class MerrorStruct

Response structure. Fields:

- Rms: float or bool: numeric | ON | OFF Range: 0 % to 100 %, Unit: % ON | OFF enables or disables the limit check.
- Peak: float or bool: numeric | ON | OFF Range: 0 % to 100 %, Unit: % ON | OFF enables or disables the limit check.

get() → MerrorStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:LIMit:MERRor
value: MerrorStruct = driver.configure.prach.limit.merror.get()
```

Defines upper limits for the RMS and peak values of the magnitude error.

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

set(*rms: float*, *peak: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:LIMit:MERRor
driver.configure.prach.limit.merror.set(rms = 1.0, peak = 1.0)
```

Defines upper limits for the RMS and peak values of the magnitude error.

param rms
(float or boolean) numeric | ON | OFF Range: 0 % to 100 %, Unit: % ON | OFF enables or disables the limit check.

param peak
(float or boolean) numeric | ON | OFF Range: 0 % to 100 %, Unit: % ON | OFF enables or disables the limit check.

6.1.7.1.3 Pdynamics

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:PRACh:LIMit:PDYNamics
```

class PdynamicsCls

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

class PdynamicsStruct

Response structure. Fields:

- Enable: bool: OFF | ON OFF: disables the limit check ON: enables the limit check

- On_Power_Upper: float: numeric Upper limit for the ON power Range: -256 dBm to 256 dBm, Unit: dBm
- On_Power_Lower: float: numeric Lower limit for the ON power Range: -256 dBm to 256 dBm, Unit: dBm
- Off_Power_Upper: float: numeric Upper limit for the OFF power Range: -256 dBm to 256 dBm, Unit: dBm

get() → PdynamicsStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:LIMit:PDYNamics
value: PdynamicsStruct = driver.configure.prach.limit.pdynamics.get()
```

Defines limits for the ON power and OFF power determined with the power dynamics measurement.

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

set(*enable: bool*, *on_power_upper: float*, *on_power_lower: float*, *off_power_upper: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:LIMit:PDYNamics
driver.configure.prach.limit.pdynamics.set(enable = False, on_power_upper = 1.0,
↪ on_power_lower = 1.0, off_power_upper = 1.0)
```

Defines limits for the ON power and OFF power determined with the power dynamics measurement.

param enable
OFF | ON OFF: disables the limit check ON: enables the limit check

param on_power_upper
numeric Upper limit for the ON power Range: -256 dBm to 256 dBm, Unit: dBm

param on_power_lower
numeric Lower limit for the ON power Range: -256 dBm to 256 dBm, Unit: dBm

param off_power_upper
numeric Upper limit for the OFF power Range: -256 dBm to 256 dBm, Unit: dBm

6.1.7.1.4 Perror

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:PRACh:LIMit:PERRor
```

class PerrorCls

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

class PerrorStruct

Response structure. Fields:

- Rms: float or bool: numeric | ON | OFF Range: 0 deg to 180 deg, Unit: deg ON | OFF enables or disables the limit check.
- Peak: float or bool: numeric | ON | OFF Range: 0 deg to 180 deg, Unit: deg ON | OFF enables or disables the limit check.

get() → PerrorStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:LIMit:PERRor
value: PerrorStruct = driver.configure.prach.limit.perror.get()
```

Defines symmetric limits for the RMS and peak values of the phase error. The limit check fails if the absolute value of the measured phase error exceeds the specified values.

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

set(*rms: float*, *peak: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:LIMit:PERRor
driver.configure.prach.limit.perror.set(rms = 1.0, peak = 1.0)
```

Defines symmetric limits for the RMS and peak values of the phase error. The limit check fails if the absolute value of the measured phase error exceeds the specified values.

param rms
(float or boolean) numeric | ON | OFF Range: 0 deg to 180 deg, Unit: deg ON | OFF enables or disables the limit check.

param peak
(float or boolean) numeric | ON | OFF Range: 0 deg to 180 deg, Unit: deg ON | OFF enables or disables the limit check.

6.1.7.2 Modulation

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:PRACh:MODulation:LRSindex
CONFigure:LTE:MEASurement<Instance>:PRACh:MODulation:ZCZConfig
CONFigure:LTE:MEASurement<Instance>:PRACh:MODulation:EWPosition
```

class ModulationCls

Modulation commands group definition. 7 total commands, 2 Subgroups, 3 group commands

get_ew_position() → LowHigh

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:MODulation:EWPosition
value: enums.LowHigh = driver.configure.prach.modulation.get_ew_position()
```

Specifies the position of the EVM window used for calculation of the trace results.

return
evm_window_pos: LOW | HIGH

get_lrs_index() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:MODulation:LRSindex
value: int = driver.configure.prach.modulation.get_lrs_index()
```

Specifies the logical root sequence index to be used for generation of the preamble sequence. For the combined signal path scenario, use CONFigure:LTE:SIGN<i>:CELL:PRACh:LRSindex.

return
log_root_seq_index: numeric Range: 0 to 837 (for preamble format 4: 0 to 137)

get_zcz_config() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:MODulation:ZCZConfig
value: int = driver.configure.prach.modulation.get_zcz_config()
```

Specifies the zero correlation zone config, i.e. which NCS value of an NCS set is used for generation of the preamble sequence. For the combined signal path scenario, use CONFigure:LTE:SIGN<i>:CELL:PRACh:ZCZConfig.

return
zero_corr_zone_con: numeric Range: 0 to 15 (for preamble format 4: 0 to 6)

set_ew_position(*evm_window_pos: LowHigh*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:MODulation:EWPosition
driver.configure.prach.modulation.set_ew_position(evm_window_pos = enums.
↪LowHigh.HIGH)
```

Specifies the position of the EVM window used for calculation of the trace results.

param evm_window_pos
LOW | HIGH

set_lrs_index(*log_root_seq_index: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:MODulation:LRSindex
driver.configure.prach.modulation.set_lrs_index(log_root_seq_index = 1)
```

Specifies the logical root sequence index to be used for generation of the preamble sequence. For the combined signal path scenario, use CONFigure:LTE:SIGN<i>:CELL:PRACh:LRSindex.

param log_root_seq_index
numeric Range: 0 to 837 (for preamble format 4: 0 to 137)

set_zcz_config(*zero_corr_zone_con: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:MODulation:ZCZConfig
driver.configure.prach.modulation.set_zcz_config(zero_corr_zone_con = 1)
```

Specifies the zero correlation zone config, i.e. which NCS value of an NCS set is used for generation of the preamble sequence. For the combined signal path scenario, use CONFigure:LTE:SIGN<i>:CELL:PRACh:ZCZConfig.

param zero_corr_zone_con
numeric Range: 0 to 15 (for preamble format 4: 0 to 6)

Cloning the Group

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

Subgroups

6.1.7.2.1 EwLength

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:PRACh:MODulation:EWLength
```

class EwLengthCls

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

get_value() → List[int]

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:MODulation:EWLength
value: List[int] = driver.configure.prach.modulation.ewLength.get_value()
```

Specifies the EVM window length in samples for all preamble formats.

return
evm_window_length: No help available

set_value(*evm_window_length: List[int]*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:MODulation:EWLength
driver.configure.prach.modulation.ewLength.set_value(evm_window_length = [1, 2,␣
↪3])
```

Specifies the EVM window length in samples for all preamble formats.

param evm_window_length
No help available

Cloning the Group

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

Subgroups

6.1.7.2.1.1 Pformat<PreambleFormat>

RepCap Settings

```
# Range: Fmt1 .. Fmt5
rc = driver.configure.prach.modulation.ewLength.pformat.repcap_preambleFormat_get()
driver.configure.prach.modulation.ewLength.pformat.repcap_preambleFormat_set(repcap.
↪PreambleFormat.Fmt1)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:PRACh:MODulation:EWLength:PFORmat<PreambleFormat>
```

class PformatCls

Pformat commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: PreambleFormat, default value after init: PreambleFormat.Fmt1

get(*preambleFormat=PreambleFormat.Default*) → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:MODulation:EWLength:PFORmat
↪<PreambleFormat>
value: int = driver.configure.prach.modulation.ewLength.pformat.
↪get(preambleFormat = repcap.PreambleFormat.Default)
```

No command help available

param preambleFormat
optional repeated capability selector. Default value: Fmt1 (settable in the interface 'Pformat')

return
evm_window_length: No help available

set(*evm_window_length: int*, *preambleFormat=PreambleFormat.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:MODulation:EWLength:PFORmat
↪<PreambleFormat>
driver.configure.prach.modulation.ewLength.pformat.set(evm_window_length = 1,␣
↪preambleFormat = repcap.PreambleFormat.Default)
```

No command help available

param evm_window_length
No help available

param preambleFormat
optional repeated capability selector. Default value: Fmt1 (settable in the interface 'Pformat')

Cloning the Group

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

6.1.7.2.2 Sindex

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:PRACh:MODulation:SINDex:AUTO
CONFigure:LTE:MEASurement<Instance>:PRACh:MODulation:SINDex
```

class SindexCls

Sindex commands group definition. 2 total commands, 0 Subgroups, 2 group commands

get_auto() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:MODulation:SINDex:AUTO
value: bool = driver.configure.prach.modulation.sindex.get_auto()
```

Enables or disables automatic detection of the sequence index. To configure the index manually for disabled automatic detection, see method RsCmwLteMeas.Configure.Prach.Modulation.Sindex.value.

return
seq_index_auto: OFF | ON

get_value() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:MODulation:SINDex
value: int = driver.configure.prach.modulation.sindex.get_value()
```

Specifies the sequence index, i.e. which of the 64 preamble sequences of the cell is used by the UE. This setting is only relevant if automatic detection is disabled, see method RsCmwLteMeas.Configure.Prach.Modulation.Sindex.auto.

return
sequence_index: numeric Range: 0 to 63

set_auto(*seq_index_auto: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:MODulation:SINDex:AUTO
driver.configure.prach.modulation.sindex.set_auto(seq_index_auto = False)
```

Enables or disables automatic detection of the sequence index. To configure the index manually for disabled automatic detection, see method RsCmwLteMeas.Configure.Prach.Modulation.Sindex.value.

param seq_index_auto
OFF | ON

set_value(*sequence_index: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:MODulation:SINDex
driver.configure.prach.modulation.sindex.set_value(sequence_index = 1)
```

Specifies the sequence index, i.e. which of the 64 preamble sequences of the cell is used by the UE. This setting is only relevant if automatic detection is disabled, see method RsCmwLteMeas.Configure.Prach.Modulation.Sindex.auto.

param sequence_index
numeric Range: 0 to 63

6.1.7.3 PfOffset

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:PRACh:PFOFfset:AUTO
CONFigure:LTE:MEASurement<Instance>:PRACh:PFOFfset
```

class PfOffsetCls

PfOffset commands group definition. 2 total commands, 0 Subgroups, 2 group commands

get_auto() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:PFOFfset:AUTO
value: bool = driver.configure.prach.pfOffset.get_auto()
```

Enables or disables automatic detection of the PRACH frequency offset. To configure the offset manually for disabled automatic detection, see method RsCmwLteMeas.Configure.Prach.PfOffset.value.

return
prach_freq_auto: OFF | ON

get_value() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:PFOFfset
value: int = driver.configure.prach.pfOffset.get_value()
```

Specifies the PRACH frequency offset. This setting is only relevant if automatic detection is disabled, see method RsCmwLteMeas.Configure.Prach.PfOffset.auto. For the combined signal path scenario, use CONFigure:LTE:SIGN<i>:CELL:PRACh:PFOFfset.

return
prach_freq_offset: numeric Range: 0 to Total RB - 6 depending on channel bandwidth, see table below

set_auto(*prach_freq_auto: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:PFOFfset:AUTO
driver.configure.prach.pfOffset.set_auto(prach_freq_auto = False)
```

Enables or disables automatic detection of the PRACH frequency offset. To configure the offset manually for disabled automatic detection, see method RsCmwLteMeas.Configure.Prach.PfOffset.value.

param prach_freq_auto
OFF | ON

set_value(*prach_freq_offset: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:PFOFfset
driver.configure.prach.pfOffset.set_value(prach_freq_offset = 1)
```

Specifies the PRACH frequency offset. This setting is only relevant if automatic detection is disabled, see method RsCmwLteMeas.Configure.Prach.PfOffset.auto. For the combined signal path scenario, use CONFigure:LTE:SIGN<i>:CELL:PRACh:PFOFfset.

param prach_freq_offset
numeric Range: 0 to Total RB - 6 depending on channel bandwidth, see table below

6.1.7.4 Power

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:PRACh:POWer:HDMode
```

class PowerCls

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

get_hdmode() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:POWer:HDMode
value: bool = driver.configure.prach.power.get_hdmode()
```

Enables or disables the high dynamic mode for power dynamics measurements.

return
high_dynamic_mode: OFF | ON

set_hdmode(*high_dynamic_mode: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:POWer:HDMode
driver.configure.prach.power.set_hdmode(high_dynamic_mode = False)
```

Enables or disables the high dynamic mode for power dynamics measurements.

param high_dynamic_mode
OFF | ON

6.1.7.5 Result

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:PRACh:RESult[:ALL]
CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:EVMagnitude
CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:EVPReamble
CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:MERRor
CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:PERRor
CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:IQ
CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:PDYNamics
CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:PVPReamble
CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:TXM
```

class ResultCls

Result commands group definition. 9 total commands, 0 Subgroups, 9 group commands

class AllStruct

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

- Evm: bool: OFF | ON Error vector magnitude OFF: Do not evaluate results. ON: Evaluate results.
- Magnitude_Error: bool: OFF | ON
- Phase_Error: bool: OFF | ON
- Iq: bool: OFF | ON I/Q constellation diagram
- Power_Dynamics: bool: OFF | ON
- Tx_Measurement: bool: OFF | ON TX measurement statistical overview
- Evm_Vs_Preamble: bool: Optional setting parameter. OFF | ON
- Power_Vs_Preamble: bool: Optional setting parameter. OFF | ON

get_all() → AllStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:RESult[:ALL]
value: AllStruct = driver.configure.prach.result.get_all()
```

Enables or disables the evaluation of results in the PRACH measurement. This command combines all other CONFigure:LTE:MEAS<i>:PRACh:RESult... commands.

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

get_ev_magnitude() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:EVMagnitude
value: bool = driver.configure.prach.result.get_ev_magnitude()
```

Enables or disables the evaluation of results in the PRACH measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- PERRor / Phase error
- IQ / I/Q constellation diagram
- PDYNamics / Power dynamics
- TXM / TX meas. statistical overview
- EVPReamble / EVM vs preamble
- PVPReamble / Power vs preamble

For reset values, see method RsCmwLteMeas.Configure.Prach.Result.all.

return
enable: OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

get_ev_preamble() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:EVPReamble
value: bool = driver.configure.prach.result.get_ev_preamble()
```

Enables or disables the evaluation of results in the PRACH measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- PERRor / Phase error
- IQ / I/Q constellation diagram
- PDYNamics / Power dynamics
- TXM / TX meas. statistical overview
- EVPReamble / EVM vs preamble
- PVPReamble / Power vs preamble

For reset values, see method RsCmwLteMeas.Configure.Prach.Result.all.

return
enable: OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

`get_iq()` → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:IQ
value: bool = driver.configure.prach.result.get_iq()
```

Enables or disables the evaluation of results in the PRACH measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- PERRor / Phase error
- IQ / I/Q constellation diagram
- PDYNamics / Power dynamics
- TXM / TX meas. statistical overview
- EVPReamble / EVM vs preamble
- PVPReamble / Power vs preamble

For reset values, see method RsCmwLteMeas.Configure.Prach.Result.all.

return
enable: OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

`get_merror()` → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:MERRor
value: bool = driver.configure.prach.result.get_merror()
```

Enables or disables the evaluation of results in the PRACH measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error

- PERRor / Phase error
- IQ / I/Q constellation diagram
- PDYNamics / Power dynamics
- TXM / TX meas. statistical overview
- EVPReamble / EVM vs preamble
- PVPReamble / Power vs preamble

For reset values, see method RsCmwLteMeas.Configure.Prach.Result.all.

return
enable: OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

get_pdynamics() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:PDYNamics
value: bool = driver.configure.prach.result.get_pdynamics()
```

Enables or disables the evaluation of results in the PRACH measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- PERRor / Phase error
- IQ / I/Q constellation diagram
- PDYNamics / Power dynamics
- TXM / TX meas. statistical overview
- EVPReamble / EVM vs preamble
- PVPReamble / Power vs preamble

For reset values, see method RsCmwLteMeas.Configure.Prach.Result.all.

return
enable: OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

get_perror() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:PERRor
value: bool = driver.configure.prach.result.get_perror()
```

Enables or disables the evaluation of results in the PRACH measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- PERRor / Phase error
- IQ / I/Q constellation diagram
- PDYNamics / Power dynamics

- TXM / TX meas. statistical overview
- EVPReamble / EVM vs preamble
- PVPReamble / Power vs preamble

For reset values, see method RsCmwLteMeas.Configure.Prach.Result.all.

return
enable: OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

get_pv_preamble() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:PVPReamble
value: bool = driver.configure.prach.result.get_pv_preamble()
```

Enables or disables the evaluation of results in the PRACH measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- PERRor / Phase error
- IQ / I/Q constellation diagram
- PDYNamics / Power dynamics
- TXM / TX meas. statistical overview
- EVPReamble / EVM vs preamble
- PVPReamble / Power vs preamble

For reset values, see method RsCmwLteMeas.Configure.Prach.Result.all.

return
enable: OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

get_txm() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:TXM
value: bool = driver.configure.prach.result.get_txm()
```

Enables or disables the evaluation of results in the PRACH measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- PERRor / Phase error
- IQ / I/Q constellation diagram
- PDYNamics / Power dynamics
- TXM / TX meas. statistical overview
- EVPReamble / EVM vs preamble
- PVPReamble / Power vs preamble

For reset values, see method RsCmwLteMeas.Configure.Prach.Result.all.

return
enable: OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

set_all(*value: AllStruct*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:RESult[:ALL]
structure = driver.configure.prach.result.AllStruct()
structure.Evm: bool = False
structure.Magnitude_Error: bool = False
structure.Phase_Error: bool = False
structure.Iq: bool = False
structure.Power_Dynamics: bool = False
structure.Tx_Measurement: bool = False
structure.Evm_Vs_Preamble: bool = False
structure.Power_Vs_Preamble: bool = False
driver.configure.prach.result.set_all(value = structure)
```

Enables or disables the evaluation of results in the PRACH measurement. This command combines all other CONFigure:LTE:MEAS<i>:PRACh:RESult... commands.

param value
see the help for AllStruct structure arguments.

set_ev_magnitude(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:EVMagnitude
driver.configure.prach.result.set_ev_magnitude(enable = False)
```

Enables or disables the evaluation of results in the PRACH measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- PERRor / Phase error
- IQ / I/Q constellation diagram
- PDYNamics / Power dynamics
- TXM / TX meas. statistical overview
- EVPReamble / EVM vs preamble
- PVPReamble / Power vs preamble

For reset values, see method RsCmwLteMeas.Configure.Prach.Result.all.

param enable
OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

set_ev_preamble(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:EVPReamble
driver.configure.prach.result.set_ev_preamble(enable = False)
```

Enables or disables the evaluation of results in the PRACH measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- PERRor / Phase error
- IQ / I/Q constellation diagram
- PDYNamics / Power dynamics
- TXM / TX meas. statistical overview
- EVPReamble / EVM vs preamble
- PVPReamble / Power vs preamble

For reset values, see method RsCmwLteMeas.Configure.Prach.Result.all.

param enable
OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

`set_iq`(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:IQ
driver.configure.prach.result.set_iq(enable = False)
```

Enables or disables the evaluation of results in the PRACH measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- PERRor / Phase error
- IQ / I/Q constellation diagram
- PDYNamics / Power dynamics
- TXM / TX meas. statistical overview
- EVPReamble / EVM vs preamble
- PVPReamble / Power vs preamble

For reset values, see method RsCmwLteMeas.Configure.Prach.Result.all.

param enable
OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

`set_merror`(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:MERRor
driver.configure.prach.result.set_merror(enable = False)
```

Enables or disables the evaluation of results in the PRACH measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error

- PERRor / Phase error
- IQ / I/Q constellation diagram
- PDYNamics / Power dynamics
- TXM / TX meas. statistical overview
- EVPReamble / EVM vs preamble
- PVPReamble / Power vs preamble

For reset values, see method RsCmwLteMeas.Configure.Prach.Result.all.

param enable
OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

`set_pdynamics`(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:PDYNamics
driver.configure.prach.result.set_pdynamics(enable = False)
```

Enables or disables the evaluation of results in the PRACH measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- PERRor / Phase error
- IQ / I/Q constellation diagram
- PDYNamics / Power dynamics
- TXM / TX meas. statistical overview
- EVPReamble / EVM vs preamble
- PVPReamble / Power vs preamble

For reset values, see method RsCmwLteMeas.Configure.Prach.Result.all.

param enable
OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

`set_perror`(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:PERRor
driver.configure.prach.result.set_perror(enable = False)
```

Enables or disables the evaluation of results in the PRACH measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- PERRor / Phase error
- IQ / I/Q constellation diagram
- PDYNamics / Power dynamics

- TXM / TX meas. statistical overview
- EVPReamble / EVM vs preamble
- PVPReamble / Power vs preamble

For reset values, see method RsCmwLteMeas.Configure.Prach.Result.all.

param enable
OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

set_pv_preamble(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:PVPReamble
driver.configure.prach.result.set_pv_preamble(enable = False)
```

Enables or disables the evaluation of results in the PRACH measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- PERRor / Phase error
- IQ / I/Q constellation diagram
- PDYNamics / Power dynamics
- TXM / TX meas. statistical overview
- EVPReamble / EVM vs preamble
- PVPReamble / Power vs preamble

For reset values, see method RsCmwLteMeas.Configure.Prach.Result.all.

param enable
OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

set_txm(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:TXM
driver.configure.prach.result.set_txm(enable = False)
```

Enables or disables the evaluation of results in the PRACH measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- PERRor / Phase error
- IQ / I/Q constellation diagram
- PDYNamics / Power dynamics
- TXM / TX meas. statistical overview
- EVPReamble / EVM vs preamble
- PVPReamble / Power vs preamble

For reset values, see method RsCmwLteMeas.Configure.Prach.Result.all.

param enable
OFF | ON OFF: Do not evaluate results. ON: Evaluate results.

6.1.7.6 Scount

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:PRACh:SCOunt:MODulation
CONFigure:LTE:MEASurement<Instance>:PRACh:SCOunt:PDYNamics
```

class ScountCls

Scount commands group definition. 2 total commands, 0 Subgroups, 2 group commands

get_modulation() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:SCOunt:MODulation
value: int = driver.configure.prach.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: numeric Range: 1 to 1000

get_pdynamics() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:SCOunt:PDYNamics
value: int = driver.configure.prach.scount.get_pdynamics()
```

Specifies the statistic count of the measurement. The statistic count is equal to the number of measurement intervals per single shot.

return
statistic_count: numeric Range: 1 to 1000

set_modulation(*statistic_count: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:SCOunt:MODulation
driver.configure.prach.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
numeric Range: 1 to 1000

set_pdynamics(*statistic_count: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:SCOunt:PDYNamics
driver.configure.prach.scount.set_pdynamics(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
numeric Range: 1 to 1000

6.1.8 RfSettings

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:RFSettings:EATTenuation
CONFigure:LTE:MEASurement<Instance>:RFSettings:UMARgin
CONFigure:LTE:MEASurement<Instance>:RFSettings:ENPower
CONFigure:LTE:MEASurement<Instance>:RFSettings:FOFFset
CONFigure:LTE:MEASurement<Instance>:RFSettings:MLOFfset
```

class RfSettingsCls

RfSettings commands group definition. 7 total commands, 2 Subgroups, 5 group commands

get_eattenuation() → float

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:RFSettings:EATTenuation
value: float = driver.configure.rfSettings.get_eattenuation()
```

Defines an external attenuation (or gain, if the value is negative) , to be applied to the input connector.

INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:LTE:SIGN<i>:RFSettings[:PCC]:EATTenuation:INPut
- CONFigure:LTE:SIGN<i>:RFSettings:SCC<c>:EATTenuation:INPut

return

rf_input_ext_att: numeric Range: -50 dB to 90 dB, Unit: dB

get_envelope_power() → float

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:RFSettings:ENPower
value: float = driver.configure.rfSettings.get_envelope_power()
```

Sets the expected nominal power of the measured RF signal.

INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:LTE:SIGN<i>:RFSettings[:PCC]:ENPMode
- CONFigure:LTE:SIGN<i>:RFSettings[:PCC]:ENPower
- CONFigure:LTE:SIGN<i>:RFSettings:SCC<c>:ENPMode
- CONFigure:LTE:SIGN<i>:RFSettings:SCC<c>:ENPower

return

exp_nom_pow: numeric 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 data sheet. Unit: dBm

get_foffset() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:RFSettings:FOFFset
value: int = driver.configure.rfSettings.get_foffset()
```

Specifies a positive or negative frequency offset to be added to the carrier center frequency (method RsCmwLteMeas. Configure.RfSettings.Cc.Frequency.set) .

INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:LTE:SIGN<i>:RFSettings[:PCC]:FOFFset:UL
- CONFigure:LTE:SIGN<i>:RFSettings:SCC<c>:FOFFset:UL
- CONFigure:LTE:SIGN<i>:RFSettings[:PCC]:FOFFset:UL:UCSPecific

return
offset: numeric Range: -100 kHz to 100 kHz, Unit: Hz

get_ml_offset() → float

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:RFSettings:MLOFfset
value: float = driver.configure.rfSettings.get_ml_offset()
```

Varies the input level of the mixer in the analyzer path.
INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:LTE:SIGN<i>:RFSettings[:PCC]:MLOFfset
- CONFigure:LTE:SIGN<i>:RFSettings:SCC<c>:MLOFfset

return
mix_lev_offset: numeric Range: -10 dB to 10 dB, Unit: dB

get_umargin() → float

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:RFSettings:UMARgin
value: float = driver.configure.rfSettings.get_umargin()
```

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 data sheet.

INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:LTE:SIGN<i>:RFSettings[:PCC]:UMARgin
- CONFigure:LTE:SIGN<i>:RFSettings:SCC<c>:UMARgin

return
user_margin: numeric Range: 0 dB to (55 dB + external attenuation - expected nominal power) , Unit: dB

set_eattenuation(*rf_input_ext_att: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:RFSettings:EATTenuation
driver.configure.rfSettings.set_eattenuation(rf_input_ext_att = 1.0)
```

Defines an external attenuation (or gain, if the value is negative) , to be applied to the input connector.
INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:LTE:SIGN<i>:RFSettings[:PCC]:EATTenuation:INPut
- CONFigure:LTE:SIGN<i>:RFSettings:SCC<c>:EATTenuation:INPut

param rf_input_ext_att
numeric Range: -50 dB to 90 dB, Unit: dB

set_envelope_power(*exp_nom_pow: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:RFSettings:ENPower
driver.configure.rfSettings.set_envelope_power(exp_nom_pow = 1.0)
```

Sets the expected nominal power of the measured RF signal.

INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:LTE:SIGN<i>:RFSettings[:PCC]:ENPMode
- CONFigure:LTE:SIGN<i>:RFSettings[:PCC]:ENPower
- CONFigure:LTE:SIGN<i>:RFSettings:SCC<c>:ENPMode
- CONFigure:LTE:SIGN<i>:RFSettings:SCC<c>:ENPower

param exp_nom_pow
numeric 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 data sheet. Unit: dBm

set_foffset(*offset: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:RFSettings:FOFFset
driver.configure.rfSettings.set_foffset(offset = 1)
```

Specifies a positive or negative frequency offset to be added to the carrier center frequency (method RsCmwLteMeas. Configure.RfSettings.Cc.Frequency.set) .

INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:LTE:SIGN<i>:RFSettings[:PCC]:FOFFset:UL
- CONFigure:LTE:SIGN<i>:RFSettings:SCC<c>:FOFFset:UL
- CONFigure:LTE:SIGN<i>:RFSettings[:PCC]:FOFFset:UL:UCSPecific

param offset
numeric Range: -100 kHz to 100 kHz, Unit: Hz

set_ml_offset(*mix_lev_offset: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:RFSettings:MLOFfset
driver.configure.rfSettings.set_ml_offset(mix_lev_offset = 1.0)
```

Varies the input level of the mixer in the analyzer path.

INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:LTE:SIGN<i>:RFSettings[:PCC]:MLOFfset
- CONFigure:LTE:SIGN<i>:RFSettings:SCC<c>:MLOFfset

param mix_lev_offset
numeric Range: -10 dB to 10 dB, Unit: dB

set_umargin(*user_margin: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:RFSettings:UMARgin
driver.configure.rfSettings.set_umargin(user_margin = 1.0)
```

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 data sheet.

INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:LTE:SIGN<i>:RFSettings[:PCC]:UMARgin
- CONFigure:LTE:SIGN<i>:RFSettings:SCC<c>:UMARgin

param user_margin
numeric Range: 0 dB to (55 dB + external attenuation - expected nominal power) , Unit: dB

Cloning the Group

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

Subgroups

6.1.8.1 Cc<CarrierComponent>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.configure.rfSettings.cc.repcap_carrierComponent_get()
driver.configure.rfSettings.cc.repcap_carrierComponent_set(repcap.CarrierComponent.Nr1)
```

class CcCls

Cc commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability: CarrierComponent, default value after init: CarrierComponent.Nr1

Cloning the Group

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

Subgroups

6.1.8.1.1 Frequency

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:RFSettings:CC<Nr>:FREQuency
```

class FrequencyCls

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

get(*carrierComponent=CarrierComponent.Default*) → float

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:RFSettings:CC<Nr>:FREQuency
value: float = driver.configure.rfSettings.cc.frequency.get(carrierComponent =␣
↪repcap.CarrierComponent.Default)
```

Selects the center frequency of component carrier CC<no>. Without carrier aggregation, you can omit <no>. Using the unit CH, the frequency can be set via the channel number. The allowed channel number range depends on the operating band, see 'Frequency bands'.

INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:LTE:SIGN<i>:RFSettings[:PCC]:CHANnel:UL
- CONFigure:LTE:SIGN<i>:RFSettings:SCC<c>:CHANnel:UL

For the supported frequency range, see 'Frequency ranges'.

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
analyzer_freq: numeric Unit: Hz

set(*analyzer_freq: float*, *carrierComponent=CarrierComponent.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:RFSettings:CC<Nr>:FREQuency
driver.configure.rfSettings.cc.frequency.set(analyzer_freq = 1.0,␣
↪carrierComponent = repcap.CarrierComponent.Default)
```

Selects the center frequency of component carrier CC<no>. Without carrier aggregation, you can omit <no>. Using the unit CH, the frequency can be set via the channel number. The allowed channel number range depends on the operating band, see 'Frequency bands'.

INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:LTE:SIGN<i>:RFSettings[:PCC]:CHANnel:UL
- CONFigure:LTE:SIGN<i>:RFSettings:SCC<c>:CHANnel:UL

For the supported frequency range, see 'Frequency ranges'.

param analyzer_freq
numeric Unit: Hz

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

6.1.8.2 Pcc

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:RFSettings[:PCC]:FREQuency
```

class PccCls

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

get_frequency() → float

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:RFSettings[:PCC]:FREQuency
value: float = driver.configure.rfSettings.pcc.get_frequency()
```

No command help available

return
analyzer_freq: No help available

set_frequency(*analyzer_freq: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:RFSettings[:PCC]:FREQuency
driver.configure.rfSettings.pcc.set_frequency(analyzer_freq = 1.0)
```

No command help available

param analyzer_freq
No help available

6.1.9 Srs

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:SRS:TOUT
CONFigure:LTE:MEASurement<Instance>:SRS:REPetition
CONFigure:LTE:MEASurement<Instance>:SRS:SCONdition
CONFigure:LTE:MEASurement<Instance>:SRS:MOEXception
CONFigure:LTE:MEASurement<Instance>:SRS:HDMode
```

class SrsCls

Srs commands group definition. 7 total commands, 2 Subgroups, 5 group commands

get_hdmode() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:SRS:HDMode
value: bool = driver.configure.srs.get_hdmode()
```

Enables or disables the high dynamic mode for power dynamics measurements.

return
high_dynamic_mode: OFF | ON

get_mo_exception() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:SRS:MOEXception
value: bool = driver.configure.srs.get_mo_exception()
```

Specifies whether measurement results that the R&S CMW identifies as faulty or inaccurate are rejected.

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

get_repetition() → Repeat

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:SRS:REPetition
value: enums.Repeat = driver.configure.srs.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 | CONTinuous SINGleshot: Single-shot measurement CONTinuous: Continuous measurement

get_scondition() → StopCondition

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:SRS:SCONdition
value: enums.StopCondition = driver.configure.srs.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 | SLFail NONE: Continue measurement irrespective of the limit check. SLFail: Stop measurement on limit failure.

get_timeout() → float

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:SRS:TOUT
value: float = driver.configure.srs.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.

return
timeout: numeric Unit: s

set_hdmode(*high_dynamic_mode: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:SRS:HDMode
driver.configure.srs.set_hdmode(high_dynamic_mode = False)
```

Enables or disables the high dynamic mode for power dynamics measurements.

param high_dynamic_mode
OFF | ON

set_mo_exception(*meas_on_exception: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:SRS:MOEXception
driver.configure.srs.set_mo_exception(meas_on_exception = False)
```

Specifies whether measurement results that the R&S CMW identifies as faulty or inaccurate are rejected.

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

set_repetition(*repetition: Repeat*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:SRS:REPetition
driver.configure.srs.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 | CONTinuous SINGleshot: Single-shot measurement CONTinuous: Continuous measurement

set_scondition(*stop_condition: StopCondition*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:SRS:SCONdition
driver.configure.srs.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 | SLFail NONE: Continue measurement irrespective of the limit check. SLFail: Stop measurement on limit failure.

set_timeout(*timeout: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:SRS:TOUT
driver.configure.srs.set_timeout(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.

param timeout
numeric Unit: s

Cloning the Group

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

Subgroups

6.1.9.1 Limit

class LimitCls

Limit 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.srs.limit.clone()
```

Subgroups

6.1.9.1.1 Pdynamics

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:SRS:LIMit:PDYNamics
```

class PdynamicsCls

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

class PdynamicsStruct

Response structure. Fields:

- Enable: bool: OFF | ON OFF: disables the limit check ON: enables the limit check
- On_Power_Upper: float: numeric Upper limit for the ON power Range: -256 dBm to 256 dBm, Unit: dBm
- On_Power_Lower: float: numeric Lower limit for the ON power Range: -256 dBm to 256 dBm, Unit: dBm
- Off_Power_Upper: float: numeric Upper limit for the OFF power Range: -256 dBm to 256 dBm, Unit: dBm

get() → PdynamicsStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:SRS:LIMit:PDYNamics
value: PdynamicsStruct = driver.configure.srs.limit.pdynamics.get()
```

Defines limits for the ON power and OFF power determined with the power dynamics measurement.

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

set(*enable: bool*, *on_power_upper: float*, *on_power_lower: float*, *off_power_upper: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:SRS:LIMit:PDYNamics
driver.configure.srs.limit.pdynamics.set(enable = False, on_power_upper = 1.0,␣
↪on_power_lower = 1.0, off_power_upper = 1.0)
```

Defines limits for the ON power and OFF power determined with the power dynamics measurement.

param enable
OFF | ON OFF: disables the limit check ON: enables the limit check

param on_power_upper
numeric Upper limit for the ON power Range: -256 dBm to 256 dBm, Unit: dBm

param on_power_lower
numeric Lower limit for the ON power Range: -256 dBm to 256 dBm, Unit: dBm

param off_power_upper
numeric Upper limit for the OFF power Range: -256 dBm to 256 dBm, Unit: dBm

6.1.9.2 Scount

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:SRS:SCOunt:PDYNamics
```

class ScountCls

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

get_pdynamics() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:SRS:SCOunt:PDYNamics
value: int = driver.configure.srs.scount.get_pdynamics()
```

Specifies the statistic count of the measurement. The statistic count is equal to the number of measurement intervals per single shot.

return
statistic_count: numeric Number of measurement intervals Range: 1 to 1000

set_pdynamics(*statistic_count: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:SRS:SCOunt:PDYNamics
driver.configure.srs.scount.set_pdynamics(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
numeric Number of measurement intervals Range: 1 to 1000

6.2 MultiEval

SCPI Commands :

```
INITiate:LTE:MEASurement<Instance>:MEValuation
STOP:LTE:MEASurement<Instance>:MEValuation
ABORt:LTE:MEASurement<Instance>:MEValuation
```

class MultiEvalCls

MultiEval commands group definition. 546 total commands, 19 Subgroups, 3 group commands

abort(*opc_timeout_ms: int = -1*) → None

```
# SCPI: ABORt:LTE:MEASurement<Instance>:MEValuation
driver.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
↪' 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:LTE:MEASurement<Instance>:MEValuation
driver.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() → None

```
# SCPI: STOP:LTE:MEASurement<Instance>:MEValuation
driver.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
```

(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.

stop_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: STOP:LTE:MEASurement<Instance>:MEValuation
driver.multiEval.stop_with_opc()


    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.

Same as stop, but waits for the operation to complete before continuing further. Use the RsCmwLteMeas.utilities.opc_timeout_set() to set the timeout value.

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.multiEval.clone()
```

Subgroups

6.2.1 Aclr

class AclrCls

Aclr 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.multiEval.aclr.clone()
```

Subgroups

6.2.1.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:ACLR:AVERage
FETCh:LTE:MEASurement<Instance>:MEValuation:ACLR:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:ACLR:AVERage
```

class AverageCls

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

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Utra_2_Neg: enums.ResultStatus2: float ACLR for the second UTRA channel with lower frequency Unit: dB
- Utra_1_Neg: enums.ResultStatus2: float ACLR for the first UTRA channel with lower frequency Unit: dB
- Eutra_Negativ: enums.ResultStatus2: float ACLR for the first E-UTRA channel with lower frequency Unit: dB
- Eutra: enums.ResultStatus2: float Power in the allocated E-UTRA channel Unit: dBm
- Eutra_Positiv: enums.ResultStatus2: float ACLR for the first E-UTRA channel with higher frequency Unit: dB
- Utra_1_Pos: enums.ResultStatus2: float ACLR for the first UTRA channel with higher frequency Unit: dB
- Utra_2_Pos: enums.ResultStatus2: float ACLR for the second UTRA channel with higher frequency Unit: dB

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Utra_2_Neg: float: float ACLR for the second UTRA channel with lower frequency Unit: dB
- Utra_1_Neg: float: float ACLR for the first UTRA channel with lower frequency Unit: dB
- Eutra_Negativ: float: float ACLR for the first E-UTRA channel with lower frequency Unit: dB
- Eutra: float: float Power in the allocated E-UTRA channel Unit: dBm
- Eutra_Positiv: float: float ACLR for the first E-UTRA channel with higher frequency Unit: dB
- Utra_1_Pos: float: float ACLR for the first UTRA channel with higher frequency Unit: dB

- Utra_2_Pos: float: float ACLR for the second UTRA channel with higher frequency Unit: dB

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:ACLR:AVERage
value: CalculateStruct = driver.multiEval.aclr.average.calculate()
```

Returns the relative ACLR values as displayed in the table below the ACLR diagram. The current and average values can be retrieved. See also 'View Spectrum ACLR'. 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:LTE:MEASurement<Instance>:MEValuation:ACLR:AVERage
value: ResultData = driver.multiEval.aclr.average.fetch()
```

Returns the relative ACLR values as displayed in the table below the ACLR diagram. The current and average values can be retrieved. See also 'View Spectrum ACLR'. 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:LTE:MEASurement<Instance>:MEValuation:ACLR:AVERage
value: ResultData = driver.multiEval.aclr.average.read()
```

Returns the relative ACLR values as displayed in the table below the ACLR diagram. The current and average values can be retrieved. See also 'View Spectrum ACLR'. 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.

6.2.1.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:ACLR:CURRent
FETCh:LTE:MEASurement<Instance>:MEValuation:ACLR:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:ACLR:CURRent
```

class CurrentCls

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

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'

- Utra_2_Neg: enums.ResultStatus2: float ACLR for the second UTRA channel with lower frequency Unit: dB
- Utra_1_Neg: enums.ResultStatus2: float ACLR for the first UTRA channel with lower frequency Unit: dB
- Eutra_Negativ: enums.ResultStatus2: float ACLR for the first E-UTRA channel with lower frequency Unit: dB
- Eutra: enums.ResultStatus2: float Power in the allocated E-UTRA channel Unit: dBm
- Eutra_Positiv: enums.ResultStatus2: float ACLR for the first E-UTRA channel with higher frequency Unit: dB
- Utra_1_Pos: enums.ResultStatus2: float ACLR for the first UTRA channel with higher frequency Unit: dB
- Utra_2_Pos: enums.ResultStatus2: float ACLR for the second UTRA channel with higher frequency Unit: dB

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Utra_2_Neg: float: float ACLR for the second UTRA channel with lower frequency Unit: dB
- Utra_1_Neg: float: float ACLR for the first UTRA channel with lower frequency Unit: dB
- Eutra_Negativ: float: float ACLR for the first E-UTRA channel with lower frequency Unit: dB
- Eutra: float: float Power in the allocated E-UTRA channel Unit: dBm
- Eutra_Positiv: float: float ACLR for the first E-UTRA channel with higher frequency Unit: dB
- Utra_1_Pos: float: float ACLR for the first UTRA channel with higher frequency Unit: dB
- Utra_2_Pos: float: float ACLR for the second UTRA channel with higher frequency Unit: dB

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:ACLR:CURRent
value: CalculateStruct = driver.multiEval.aclr.current.calculate()
```

Returns the relative ACLR values as displayed in the table below the ACLR diagram. The current and average values can be retrieved. See also 'View Spectrum ACLR'. 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:LTE:MEASurement<Instance>:MEValuation:ACLR:CURRent
value: ResultData = driver.multiEval.aclr.current.fetch()
```

Returns the relative ACLR values as displayed in the table below the ACLR diagram. The current and average values can be retrieved. See also 'View Spectrum ACLR'. 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:LTE:MEASurement<Instance>:MEValuation:ACLR:CURRent
value: ResultData = driver.multiEval.aclr.current.read()
```

Returns the relative ACLR values as displayed in the table below the ACLR diagram. The current and average values can be retrieved. See also 'View Spectrum ACLR'. 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.

6.2.1.3 Dallocation

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:ACLR:DALLocation
```

class DallocationCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Nr_Res_Blocks: int: decimal Number of allocated resource blocks
- Offset_Res_Blocks: int: decimal Offset of the first allocated resource block from the edge of the allocated UL transmission bandwidth

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:ACLR:DALLocation
value: FetchStruct = driver.multiEval.aclr.dallocation.fetch()
```

Returns the detected allocation for the measured slot. If the same slot is measured by the individual measurements, all commands yield the same result. If different statistic counts are defined for the modulation, ACLR and spectrum emission mask measurements, different slots can be measured and different results can be returned by the individual commands.

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

6.2.1.4 DchType

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:ACLR:DCHType
```

class DchTypeCls

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

fetch() → UplinkChannelType

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:ACLR:DCHType
value: enums.UplinkChannelType = driver.multiEval.aclr.dchType.fetch()
```

Returns the uplink channel type for the measured slot. If the same slot is measured by the individual measurements, all commands yield the same result. If different statistic counts are defined for the modulation, ACLR and spectrum emission mask measurements, different slots can be measured and different results can be returned by the individual commands.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
channel_type: PUSCh | PUCCh

6.2.2 Amarker<AbsMarker>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.multiEval.amarker.repcap_absMarker_get()
driver.multiEval.amarker.repcap_absMarker_set(repcap.AbsMarker.Nr1)
```

class AmarkerCls

Amarker commands group definition. 6 total commands, 5 Subgroups, 0 group commands Repeated Capability: AbsMarker, default value after init: AbsMarker.Nr1

Cloning the Group

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

Subgroups

6.2.2.1 EvMagnitude

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:AMARker<No>:EVMagnitude
```

class EvMagnitudeCls

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

fetch(*xvalue: int*, *trace_select: TraceSelect*, *absMarker=AbsMarker.Default*) → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:AMARker<No>:EVMagnitude
value: float = driver.multiEval.amarker.evMagnitude.fetch(xvalue = 1, trace_
↪select = enums.TraceSelect.AVERage, absMarker = repcap.AbsMarker.Default)
```

Uses the markers 1 and 2 with absolute values on the bar graphs: EVM RMS, EVM peak, magnitude error and phase error vs SC-FDMA symbol.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param xvalue
(integer or boolean) integer Absolute X value of the marker position There are two X values per SC-FDMA symbol on the X axis (symbol 0 low, symbol 0 high, ..., symbol 6 low, symbol 6 high) . Range: 0 to 13

param trace_select
CURRent | AVERage | MAXimum

param absMarker
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Amarker')

return
yvalue: float Absolute Y value of the marker position

Cloning the Group

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

Subgroups

6.2.2.1.1 Peak

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:AMARker<No>:EVMagnitude:PEAK
```

class PeakCls

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

fetch(*xvalue: int*, *trace_select: TraceSelect*, *absMarker=AbsMarker.Default*) → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:AMARker<No>:EVMagnitude:PEAK
value: float = driver.multiEval.amarker.evMagnitude.peak.fetch(xvalue = 1,␣
↪trace_select = enums.TraceSelect.AVERage, absMarker = repcap.AbsMarker.
↪Default)
```

Uses the markers 1 and 2 with absolute values on the bar graphs: EVM RMS, EVM peak, magnitude error and phase error vs SC-FDMA symbol.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param xvalue
(integer or boolean) integer Absolute X value of the marker position There are two X values per SC-FDMA symbol on the X axis (symbol 0 low, symbol 0 high, ..., symbol 6 low, symbol 6 high) . Range: 0 to 13

param trace_select
CURRent | AVERage | MAXimum

param absMarker
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Amarker')

return
yvalue: float Absolute Y value of the marker position

6.2.2.2 Merror

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:AMARker<No>:MERRor
```

class MerrorCls

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

fetch(*xvalue: int*, *trace_select: TraceSelect*, *absMarker=AbsMarker.Default*) → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:AMARker<No>:MERRor
value: float = driver.multiEval.amarker.merror.fetch(xvalue = 1, trace_select =␣
↪enums.TraceSelect.AVERage, absMarker = repcap.AbsMarker.Default)
```

Uses the markers 1 and 2 with absolute values on the bar graphs: EVM RMS, EVM peak, magnitude error and phase error vs SC-FDMA symbol.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param xvalue
(integer or boolean) integer Absolute X value of the marker position There are two X values per SC-FDMA symbol on the X axis (symbol 0 low, symbol 0 high, ..., symbol 6 low, symbol 6 high) . Range: 0 to 13

param trace_select
CURRent | AVERage | MAXimum

param absMarker
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Amarker')

return
yvalue: float Absolute Y value of the marker position

6.2.2.3 Pdynamics

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:AMARker<No>:PDYNamics
```

class PdynamicsCls

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

fetch(*xvalue: float*, *trace_select: TraceSelect*, *absMarker=AbsMarker.Default*) → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:AMARker<No>:PDYNamics
value: float = driver.multiEval.amarker.pdynamics.fetch(xvalue = 1.0, trace_
↪select = enums.TraceSelect.AVERage, absMarker = repcap.AbsMarker.Default)
```

Uses the markers 1 and 2 with absolute values on the power dynamics trace.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param xvalue
(float or boolean) integer Absolute X value of the marker position Range: -1100 µs to 2500 µs, Unit: µs

param trace_select
CURRent | AVERage | MAXimum

param absMarker
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Amarker')

return
yvalue: float Absolute Y value of the marker position Unit: dBm

6.2.2.4 Perror

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:AMARker<No>:PERRor
```

class PerrorCls

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

fetch(*xvalue: int*, *trace_select: TraceSelect*, *absMarker=AbsMarker.Default*) → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:AMARker<No>:PERRor
value: float = driver.multiEval.amarker.perror.fetch(xvalue = 1, trace_select =␣
↪enums.TraceSelect.AVERage, absMarker = repcap.AbsMarker.Default)
```

Uses the markers 1 and 2 with absolute values on the bar graphs: EVM RMS, EVM peak, magnitude error and phase error vs SC-FDMA symbol.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param xvalue
(integer or boolean) integer Absolute X value of the marker position There are two X values per SC-FDMA symbol on the X axis (symbol 0 low, symbol 0 high, . . . , symbol 6 low, symbol 6 high) . Range: 0 to 13

param trace_select
CURRent | AVERage | MAXimum

param absMarker
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Amarker')

return
yvalue: float Absolute Y value of the marker position

6.2.2.5 Pmonitor

class PmonitorCls

Pmonitor 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.multiEval.amarker.pmonitor.clone()
```

Subgroups

6.2.2.5.1 Cc<CarrierComponent>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.multiEval.amarker.pmonitor.cc.repcap_carrierComponent_get()
driver.multiEval.amarker.pmonitor.cc.repcap_carrierComponent_set(repcap.CarrierComponent.
↪Nr1)
```

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:AMARker<No>:PMONitor:CC<Nr>
```

class CcCls

Cc commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: CarrierComponent, default value after init: CarrierComponent.Nr1

fetch(*xvalue: int*, *absMarker=AbsMarker.Default*, *carrierComponent=CarrierComponent.Default*) → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:AMARker<No>:PMONitor:CC<Nr>
value: float = driver.multiEval.amarker.pmonitor.cc.fetch(xvalue = 1, absMarker␣
↪= repcap.AbsMarker.Default, carrierComponent = repcap.CarrierComponent.
↪Default)
```

Uses the markers 1 and 2 with absolute values on the power monitor trace.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param xvalue
(integer or boolean) integer Absolute X value of the marker position (subframe number) Range: 0 to 319

param absMarker
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Amarker')

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
yvalue: float Absolute Y value of the marker position Unit: dBm

Cloning the Group

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

6.2.3 Bler

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:BLER
READ:LTE:MEASurement<Instance>:MEValuation:BLER
```

class BlerCls

Bler commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Ack: float: float Received acknowledgments (percentage of sent scheduled subframes) . Unit: %
- Nack: float: float Received negative acknowledgments (percentage of sent scheduled subframes) . Unit: %
- Bler: float: float Block error ratio (percentage of sent scheduled subframes for which no ACK has been received) . Unit: %
- Dtx: float: float Percentage of sent scheduled subframes for which no ACK and no NACK has been received. Unit: %

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:BLER
value: ResultData = driver.multiEval.bler.fetch()
```

Returns the block error ratio results determined from all captured subframes.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:BLER
value: ResultData = driver.multiEval.bler.read()
```

Returns the block error ratio results determined from all captured subframes.

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

6.2.4 Dmarker<DeltaMarker>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.multiEval.dmarker.repcap_deltaMarker_get()
driver.multiEval.dmarker.repcap_deltaMarker_set(repcap.DeltaMarker.Nr1)
```

class DmarkerCls

Dmarker commands group definition. 6 total commands, 5 Subgroups, 0 group commands Repeated Capability: DeltaMarker, default value after init: DeltaMarker.Nr1

Cloning the Group

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

Subgroups

6.2.4.1 EvMagnitude

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:DMARker<No>:EVMagnitude
```

class EvMagnitudeCls

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

fetch(*xvalue: int*, *trace_select: TraceSelect*, *deltaMarker=DeltaMarker.Default*) → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:DMARker<No>:EVMagnitude
value: float = driver.multiEval.dmarker.evMagnitude.fetch(xvalue = 1, trace_
↪select = enums.TraceSelect.AVERage, deltaMarker = repcap.DeltaMarker.Default)
```

Uses the markers 1 and 2 with relative values on the bar graphs: EVM RMS, EVM peak, magnitude error and phase error vs SC-FDMA symbol.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param xvalue
(integer or boolean) integer X value of the marker position relative to the X value of the reference marker There are two X values per SC-FDMA symbol on the X axis (symbol 0 low, symbol 0 high, ..., symbol 6 low, symbol 6 high) . Range: -13 to 13

param trace_select
CURRent | AVERage | MAXimum

param deltaMarker
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Dmarker')

return
yvalue: float Y value of the marker position relative to the Y value of the reference marker

Cloning the Group

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

Subgroups

6.2.4.1.1 Peak

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:DMARker<No>:EVMagnitude:PEAK
```

class PeakCls

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

fetch(*xvalue: int*, *trace_select: TraceSelect*, *deltaMarker=DeltaMarker.Default*) → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:DMARker<No>:EVMagnitude:PEAK
value: float = driver.multiEval.dmarker.evMagnitude.peak.fetch(xvalue = 1,␣
↪trace_select = enums.TraceSelect.AVERage, deltaMarker = repcap.DeltaMarker.
↪Default)
```

Uses the markers 1 and 2 with relative values on the bar graphs: EVM RMS, EVM peak, magnitude error and phase error vs SC-FDMA symbol.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param xvalue
(integer or boolean) integer X value of the marker position relative to the X value of the reference marker There are two X values per SC-FDMA symbol on the X axis (symbol 0 low, symbol 0 high, ..., symbol 6 low, symbol 6 high) . Range: -13 to 13

param trace_select
CURRent | AVERage | MAXimum

param deltaMarker
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Dmarker')

return
yvalue: float Y value of the marker position relative to the Y value of the reference marker

6.2.4.2 Merror

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:DMARker<No>:MERRor
```

class MerrorCls

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

fetch(*xvalue: int*, *trace_select: TraceSelect*, *deltaMarker=DeltaMarker.Default*) → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:DMARker<No>:MERRor
value: float = driver.multiEval.dmarker.merror.fetch(xvalue = 1, trace_select =␣
↪enums.TraceSelect.AVERage, deltaMarker = repcap.DeltaMarker.Default)
```

Uses the markers 1 and 2 with relative values on the bar graphs: EVM RMS, EVM peak, magnitude error and phase error vs SC-FDMA symbol.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param xvalue
(integer or boolean) integer X value of the marker position relative to the X value of the reference marker There are two X values per SC-FDMA symbol on the X axis (symbol 0 low, symbol 0 high, ..., symbol 6 low, symbol 6 high) . Range: -13 to 13

param trace_select
CURRent | AVERage | MAXimum

param deltaMarker
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Dmarker')

return
yvalue: float Y value of the marker position relative to the Y value of the reference marker

6.2.4.3 Pdynamics

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:DMARker<No>:PDYNamics
```

class PdynamicsCls

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

fetch(*xvalue: float*, *trace_select: TraceSelect*, *deltaMarker=DeltaMarker.Default*) → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:DMARker<No>:PDYNamics
value: float = driver.multiEval.dmarker.pdynamics.fetch(xvalue = 1.0, trace_
↪select = enums.TraceSelect.AVERage, deltaMarker = repcap.DeltaMarker.Default)
```

Uses the markers 1 and 2 with relative values on the power dynamics trace.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param xvalue
(float or boolean) integer X value of the marker position relative to the X value of the reference marker Range: -3600 µs to 3600 µs, Unit: µs

param trace_select
CURRent | AVERage | MAXimum

param deltaMarker
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Dmarker')

return
yvalue: float Y value of the marker position relative to the Y value of the reference marker Unit: dB

6.2.4.4 Perror

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:DMARker<No>:PERRor
```

class PerrorCls

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

fetch(*xvalue: int*, *trace_select: TraceSelect*, *deltaMarker=DeltaMarker.Default*) → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:DMARker<No>:PERRor
value: float = driver.multiEval.dmarker.perror.fetch(xvalue = 1, trace_select =␣
↪enums.TraceSelect.AVERage, deltaMarker = repcap.DeltaMarker.Default)
```

Uses the markers 1 and 2 with relative values on the bar graphs: EVM RMS, EVM peak, magnitude error and phase error vs SC-FDMA symbol.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param xvalue
(integer or boolean) integer X value of the marker position relative to the X value of the reference marker There are two X values per SC-FDMA symbol on the X axis (symbol 0 low, symbol 0 high, ..., symbol 6 low, symbol 6 high) . Range: -13 to 13

param trace_select
CURRent | AVERage | MAXimum

param deltaMarker
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Dmarker')

return
yvalue: float Y value of the marker position relative to the Y value of the reference marker

6.2.4.5 Pmonitor

class PmonitorCls

Pmonitor 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.multiEval.dmarker.pmonitor.clone()
```

Subgroups

6.2.4.5.1 Cc<CarrierComponent>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.multiEval.dmarker.pmonitor.cc.repcap_carrierComponent_get()
driver.multiEval.dmarker.pmonitor.cc.repcap_carrierComponent_set(repcap.CarrierComponent.
↪Nr1)
```

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:DMARker<No>:PMONitor:CC<Nr>
```

class CcCls

Cc commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: CarrierComponent, default value after init: CarrierComponent.Nr1

fetch(*xvalue: int*, *deltaMarker=DeltaMarker.Default*, *carrierComponent=CarrierComponent.Default*) → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:DMARker<No>:PMONitor:CC<Nr>
value: float = driver.multiEval.dmarker.pmonitor.cc.fetch(xvalue = 1,␣
↪deltaMarker = repcap.DeltaMarker.Default, carrierComponent = repcap.
↪CarrierComponent.Default)
```

Uses the markers 1 and 2 with relative values on the power monitor trace.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param xvalue
(integer or boolean) integer X value of the marker position relative to the X value of the reference marker (in subframes) Range: -319 to 319

param deltaMarker
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Dmarker')

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
yvalue: float Y value of the marker position relative to the Y value of the reference marker Unit: dB

Cloning the Group

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

6.2.5 EsFlatness

class EsFlatnessCls

EsFlatness 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.multiEval.esFlatness.clone()
```

Subgroups

6.2.5.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:ESFLatness:AVERage
FETCh:LTE:MEASurement<Instance>:MEValuation:ESFLatness:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:ESFLatness:AVERage
```

class AverageCls

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

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits. Unit: %
- Ripple_1: float or bool: Limit check result for max (range 1) - min (range 1) .
- Ripple_2: float or bool: Limit check result for max (range 2) - min (range 2) .
- Max_R_1_Min_R_2: float or bool: Limit check result for max (range 1) - min (range 2) .
- Max_R_2_Min_R_1: float or bool: Limit check result for max (range 2) - min (range 1) .

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'

- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits. Unit: %
- Ripple_1: float: float Max (range 1) - min (range 1) Unit: dB
- Ripple_2: float: float Max (range 2) - min (range 2) Unit: dB
- Max_R_1_Min_R_2: float: float Max (range 1) - min (range 2) Unit: dB
- Max_R_2_Min_R_1: float: float Max (range 2) - min (range 1) Unit: dB
- Min_R_1: float: float Min (range 1) Unit: dB
- Max_R_1: float: float Max (range 1) Unit: dB
- Min_R_2: float: float Min (range 2) Unit: dB
- Max_R_2: float: float Max (range 2) Unit: dB

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:ESFLatness:AVERage
value: CalculateStruct = driver.multiEval.esFlatness.average.calculate()
```

Return current, average and extreme single-value results of the equalizer spectrum flatness measurement. See also 'Equalizer spectrum flatness limits'.

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

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:ESFLatness:AVERage
value: ResultData = driver.multiEval.esFlatness.average.fetch()
```

Return current, average, extreme and standard deviation single-value results of the equalizer spectrum flatness measurement. See also 'Equalizer spectrum flatness limits'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:ESFLatness:AVERage
value: ResultData = driver.multiEval.esFlatness.average.read()
```

Return current, average, extreme and standard deviation single-value results of the equalizer spectrum flatness measurement. See also 'Equalizer spectrum flatness limits'.

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

6.2.5.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:ESFLatness:CURRent
FETCh:LTE:MEASurement<Instance>:MEValuation:ESFLatness:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:ESFLatness:CURRent
```

class CurrentCls

Current commands group definition. 4 total commands, 1 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits. Unit: %
- Ripple_1: float or bool: Limit check result for max (range 1) - min (range 1) .
- Ripple_2: float or bool: Limit check result for max (range 2) - min (range 2) .
- Max_R_1_Min_R_2: float or bool: Limit check result for max (range 1) - min (range 2) .
- Max_R_2_Min_R_1: float or bool: Limit check result for max (range 2) - min (range 1) .

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits. Unit: %
- Ripple_1: float: float Max (range 1) - min (range 1) Unit: dB
- Ripple_2: float: float Max (range 2) - min (range 2) Unit: dB
- Max_R_1_Min_R_2: float: float Max (range 1) - min (range 2) Unit: dB
- Max_R_2_Min_R_1: float: float Max (range 2) - min (range 1) Unit: dB
- Min_R_1: float: float Min (range 1) Unit: dB
- Max_R_1: float: float Max (range 1) Unit: dB
- Min_R_2: float: float Min (range 2) Unit: dB
- Max_R_2: float: float Max (range 2) Unit: dB

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:ESFLatness:CURRent
value: CalculateStruct = driver.multiEval.esFlatness.current.calculate()
```

Return current, average and extreme single-value results of the equalizer spectrum flatness measurement. See also 'Equalizer spectrum flatness limits'.

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

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:ESFLatness:CURRent
value: ResultData = driver.multiEval.esFlatness.current.fetch()
```

Return current, average, extreme and standard deviation single-value results of the equalizer spectrum flatness measurement. See also 'Equalizer spectrum flatness limits'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:ESFLatness:CURRent
value: ResultData = driver.multiEval.esFlatness.current.read()
```

Return current, average, extreme and standard deviation single-value results of the equalizer spectrum flatness measurement. See also 'Equalizer spectrum flatness limits'.

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.multiEval.esFlatness.current.clone()
```

Subgroups

6.2.5.2.1 ScIndex

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:ESFLatness:CURRent:SCINdex
```

class ScIndexCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits. Unit: %
- Maximum_1: int: decimal SC index of 'Max (Range 1) '
- Minimum_1: int: decimal SC index of 'Min (Range 1) '
- Maximum_2: int: decimal SC index of 'Max (Range 2) '
- Minimum_2: int: decimal SC index of 'Min (Range 2) '

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:ESFLatness:CURRent:SCINdex
value: FetchStruct = driver.multiEval.esFlatness.current.scIndex.fetch()
```

Returns subcarrier indices of the equalizer spectrum flatness measurement. At these SC indices, the current minimum and maximum power of the equalizer coefficients have been detected within range 1 and range 2.

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

6.2.5.3 Extreme

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:ESFLatness:EXTReme
FETCh:LTE:MEASurement<Instance>:MEValuation:ESFLatness:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:ESFLatness:EXTReme
```

class ExtremeCls

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

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits. Unit: %
- Ripple_1: float or bool: Limit check result for max (range 1) - min (range 1) .
- Ripple_2: float or bool: Limit check result for max (range 2) - min (range 2) .
- Max_R_1_Min_R_2: float or bool: Limit check result for max (range 1) - min (range 2) .
- Max_R_2_Min_R_1: float or bool: Limit check result for max (range 2) - min (range 1) .

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits. Unit: %
- Ripple_1: float: float Max (range 1) - min (range 1) Unit: dB
- Ripple_2: float: float Max (range 2) - min (range 2) Unit: dB
- Max_R_1_Min_R_2: float: float Max (range 1) - min (range 2) Unit: dB
- Max_R_2_Min_R_1: float: float Max (range 2) - min (range 1) Unit: dB
- Min_R_1: float: float Min (range 1) Unit: dB
- Max_R_1: float: float Max (range 1) Unit: dB
- Min_R_2: float: float Min (range 2) Unit: dB
- Max_R_2: float: float Max (range 2) Unit: dB

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:ESFLatness:EXTReme
value: CalculateStruct = driver.multiEval.esFlatness.extreme.calculate()
```

Return current, average and extreme single-value results of the equalizer spectrum flatness measurement. See also 'Equalizer spectrum flatness limits'.

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

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:ESFLatness:EXTReme
value: ResultData = driver.multiEval.esFlatness.extreme.fetch()
```

Return current, average, extreme and standard deviation single-value results of the equalizer spectrum flatness measurement. See also 'Equalizer spectrum flatness limits'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:ESFLatness:EXTReme
value: ResultData = driver.multiEval.esFlatness.extreme.read()
```

Return current, average, extreme and standard deviation single-value results of the equalizer spectrum flatness measurement. See also 'Equalizer spectrum flatness limits'.

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

6.2.5.4 StandardDev

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:ESFLatness:SDEViation
FETCh:LTE:MEASurement<Instance>:MEValuation:ESFLatness:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits. Unit: %
- Ripple_1: float: float Max (range 1) - min (range 1) Unit: dB
- Ripple_2: float: float Max (range 2) - min (range 2) Unit: dB
- Max_R_1_Min_R_2: float: float Max (range 1) - min (range 2) Unit: dB
- Max_R_2_Min_R_1: float: float Max (range 2) - min (range 1) Unit: dB

- Min_R_1: float: float Min (range 1) Unit: dB
- Max_R_1: float: float Max (range 1) Unit: dB
- Min_R_2: float: float Min (range 2) Unit: dB
- Max_R_2: float: float Max (range 2) Unit: dB

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:ESFLatness:SDEViation
value: ResultData = driver.multiEval.esFlatness.standardDev.fetch()
```

Return current, average, extreme and standard deviation single-value results of the equalizer spectrum flatness measurement. See also 'Equalizer spectrum flatness limits'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:ESFLatness:SDEViation
value: ResultData = driver.multiEval.esFlatness.standardDev.read()
```

Return current, average, extreme and standard deviation single-value results of the equalizer spectrum flatness measurement. See also 'Equalizer spectrum flatness limits'.

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

6.2.6 EvMagnitude

class EvMagnitudeCls

EvMagnitude 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.multiEval.evMagnitude.clone()
```

Subgroups

6.2.6.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:AVERage
FETCh:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Low: List[float]: float EVM value for low EVM window position Unit: %
- High: List[float]: float EVM value for high EVM window position Unit: %

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:AVERage
value: ResultData = driver.multiEval.evMagnitude.average.fetch()
```

Returns the values of the EVM RMS bar graphs for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum bar graphs can be retrieved. There is one pair of EVM values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, … See also 'View EVM'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:AVERage
value: ResultData = driver.multiEval.evMagnitude.average.read()
```

Returns the values of the EVM RMS bar graphs for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum bar graphs can be retrieved. There is one pair of EVM values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, … See also 'View EVM'.

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

6.2.6.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:CURRent
FETCh:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Low: List[float]: float EVM value for low EVM window position Unit: %
- High: List[float]: float EVM value for high EVM window position Unit: %

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:CURRent
value: ResultData = driver.multiEval.evMagnitude.current.fetch()
```

Returns the values of the EVM RMS bar graphs for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum bar graphs can be retrieved. There is one pair of EVM values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'View EVM'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:CURRent
value: ResultData = driver.multiEval.evMagnitude.current.read()
```

Returns the values of the EVM RMS bar graphs for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum bar graphs can be retrieved. There is one pair of EVM values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'View EVM'.

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

6.2.6.3 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:MAXimum
FETCh:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Low: List[float]: float EVM value for low EVM window position Unit: %
- High: List[float]: float EVM value for high EVM window position Unit: %

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:MAXimum
value: ResultData = driver.multiEval.evMagnitude.maximum.fetch()
```

Returns the values of the EVM RMS bar graphs for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum bar graphs can be retrieved. There is one pair of EVM values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'View EVM'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:MAXimum
value: ResultData = driver.multiEval.evMagnitude.maximum.read()
```

Returns the values of the EVM RMS bar graphs for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum bar graphs can be retrieved. There is one pair of EVM values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'View EVM'.

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

6.2.6.4 Peak

class PeakCls

Peak 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.multiEval.evMagnitude.peak.clone()
```

Subgroups

6.2.6.4.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:PEAK:AVERage
FETCh:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:PEAK:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Low: List[float]: float EVM value for low EVM window position Unit: %
- High: List[float]: float EVM value for high EVM window position Unit: %

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:PEAK:AVERage
value: ResultData = driver.multiEval.evMagnitude.peak.average.fetch()
```

Returns the values of the EVM peak bar graphs for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum bar graphs can be retrieved. There is one pair of EVM values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'View EVM'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:PEAK:AVERage
value: ResultData = driver.multiEval.evMagnitude.peak.average.read()
```

Returns the values of the EVM peak bar graphs for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum bar graphs can be retrieved. There is one pair of EVM values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'View EVM'.

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

6.2.6.4.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:PEAK:CURRent
FETCh:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:PEAK:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Low: List[float]: float EVM value for low EVM window position Unit: %
- High: List[float]: float EVM value for high EVM window position Unit: %

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:PEAK:CURRent
value: ResultData = driver.multiEval.evMagnitude.peak.current.fetch()
```

Returns the values of the EVM peak bar graphs for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum bar graphs can be retrieved. There is one pair of EVM values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'View EVM'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:PEAK:CURRent
value: ResultData = driver.multiEval.evMagnitude.peak.current.read()
```

Returns the values of the EVM peak bar graphs for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum bar graphs can be retrieved. There is one pair of EVM values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'View EVM'.

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

6.2.6.4.3 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:PEAK:MAXimum
FETCh:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:PEAK:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Low: List[float]: float EVM value for low EVM window position Unit: %
- High: List[float]: float EVM value for high EVM window position Unit: %

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:PEAK:MAXimum
value: ResultData = driver.multiEval.evMagnitude.peak.maximum.fetch()
```

Returns the values of the EVM peak bar graphs for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum bar graphs can be retrieved. There is one pair of EVM values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'View EVM'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:PEAK:MAXimum
value: ResultData = driver.multiEval.evMagnitude.peak.maximum.read()
```

Returns the values of the EVM peak bar graphs for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum bar graphs can be retrieved. There is one pair of EVM values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'View EVM'.

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

6.2.7 Evmc

class EvmcCls

Evmc commands group definition. 8 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.7.1 Peak

class PeakCls

Peak 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.multiEval.evmc.peak.clone()
```

Subgroups

6.2.7.1.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:EVMC:PEAK:AVERage
FETCh:LTE:MEASurement<Instance>:MEValuation:EVMC:PEAK:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:EVMC:PEAK:AVERage
value: float = driver.multiEval.evmc.peak.average.fetch()
```

The CURRent command returns the maximum value of the EVM vs subcarrier trace. The AVERage, MAXimum and SDEViation values are calculated from the CURRent values. The peak results cannot be displayed at the GUI.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_cpeak_average: float Unit: %

read() → float

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:EVMC:PEAK:AVERage
value: float = driver.multiEval.evmc.peak.average.read()
```

The CURRent command returns the maximum value of the EVM vs subcarrier trace. The AVERage, MAXimum and SDEViation values are calculated from the CURRent values. The peak results cannot be displayed at the GUI.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_cpeak_average: float Unit: %

6.2.7.1.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:EVMC:PEAK:CURRent
FETCh:LTE:MEASurement<Instance>:MEValuation:EVMC:PEAK:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:EVMC:PEAK:CURRent
value: float = driver.multiEval.evmc.peak.current.fetch()
```

The CURRent command returns the maximum value of the EVM vs subcarrier trace. The AVERage, MAXimum and SDEViation values are calculated from the CURRent values. The peak results cannot be displayed at the GUI.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_cpeak_current: float Unit: %

read() → float

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:EVMC:PEAK:CURRent
value: float = driver.multiEval.evmc.peak.current.read()
```

The CURRent command returns the maximum value of the EVM vs subcarrier trace. The AVERage, MAXimum and SDEViation values are calculated from the CURRent values. The peak results cannot be displayed at the GUI.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_cpeak_current: float Unit: %

6.2.7.1.3 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:EVMC:PEAK:MAXimum
FETCh:LTE:MEASurement<Instance>:MEValuation:EVMC:PEAK:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:EVMC:PEAK:MAXimum
value: float = driver.multiEval.evmc.peak.maximum.fetch()
```

The CURRent command returns the maximum value of the EVM vs subcarrier trace. The AVERage, MAXimum and SDEViation values are calculated from the CURRent values. The peak results cannot be displayed at the GUI.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_cpeak_maximum: float Unit: %

read() → float

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:EVMC:PEAK:MAXimum
value: float = driver.multiEval.evmc.peak.maximum.read()
```

The CURRent command returns the maximum value of the EVM vs subcarrier trace. The AVERage, MAXimum and SDEViation values are calculated from the CURRent values. The peak results cannot be displayed at the GUI.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_cpeak_maximum: float Unit: %

6.2.7.1.4 StandardDev

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:EVMC:PEAK:SDEViation
FETCh:LTE:MEASurement<Instance>:MEValuation:EVMC:PEAK:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:EVMC:PEAK:SDEViation
value: float = driver.multiEval.evmc.peak.standardDev.fetch()
```

The CURRent command returns the maximum value of the EVM vs subcarrier trace. The AVERage, MAXimum and SDEViation values are calculated from the CURRent values. The peak results cannot be displayed at the GUI.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_cpeak_std_dev: float Unit: %

read() → float

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:EVMC:PEAK:SDEViation
value: float = driver.multiEval.evmc.peak.standardDev.read()
```

The CURRent command returns the maximum value of the EVM vs subcarrier trace. The AVERage, MAXimum and SDEViation values are calculated from the CURRent values. The peak results cannot be displayed at the GUI.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_cpeak_std_dev: float Unit: %

6.2.8 InbandEmission

class InbandEmissionCls

InbandEmission 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.multiEval.inbandEmission.clone()
```

Subgroups

6.2.8.1 Cc<CarrierComponent>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.multiEval.inbandEmission.cc.repcap_carrierComponent_get()
driver.multiEval.inbandEmission.cc.repcap_carrierComponent_set(repcap.CarrierComponent.
↪Nr1)
```

class CcCls

Cc commands group definition. 6 total commands, 1 Subgroups, 0 group commands Repeated Capability: CarrierComponent, default value after init: CarrierComponent.Nr1

Cloning the Group

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

Subgroups

6.2.8.1.1 Margin

class MarginCls

Margin commands group definition. 6 total commands, 4 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.8.1.1.1 Average

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:CC<Nr>:MARGin:AVERage
```

class AverageCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits. Unit: %
- Margin: float: float Unit: dB

fetch(*carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:CC<Nr>
↪:MARGin:AVERage
value: FetchStruct = driver.multiEval.inbandEmission.cc.margin.average.
↪fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return the limit line margin results for the CC<no> diagram. The CURRent margin indicates the minimum (vertical) distance between the inband emissions limit line and the current trace. A negative result indicates that the limit is exceeded. The AVERage, EXTReme and SDEViation values are calculated from the current margins. The margin results cannot be displayed at the GUI.

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

6.2.8.1.1.2 Current

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:CC<Nr>:MARGin:CURRent
```

class CurrentCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits. Unit: %
- Margin: float: float Unit: dB

fetch(*carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:CC<Nr>
↪:MARGin:CURRent
value: FetchStruct = driver.multiEval.inbandEmission.cc.margin.current.
↪fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return the limit line margin results for the CC<no> diagram. The CURRent margin indicates the minimum (vertical) distance between the inband emissions limit line and the current trace. A negative result indicates that the limit is exceeded. The AVERage, EXTReme and SDEViation values are calculated from the current margins. The margin results cannot be displayed at the GUI.

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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.multiEval.inbandEmission.cc.margin.current.clone()
```

Subgroups

6.2.8.1.1.3 RbIndex

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:CC<Nr>:MARGin:CURRent:RBINdex
```

class RbIndexCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits. Unit: %
- Rb_Index: int: decimal Resource block index

fetch(*carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:CC<Nr>
↪:MARGin:CURRent:RBINdex
value: FetchStruct = driver.multiEval.inbandEmission.cc.margin.current.rbIndex.
↪fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return resource block indices for CC<no> inband emission margins. At these RB indices, the CURRent and EXTReme margins have been detected (see method RsCmwLteMeas.MultiEval.InbandEmission.Cc.Margin.Current.fetch and . . . :EXTReme) .

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

6.2.8.1.1.4 Extreme

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:CC<Nr>:MARGin:EXTReme
```

class ExtremeCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits. Unit: %
- Margin: float: float Unit: dB

fetch(*carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:CC<Nr>
↪:MARGin:EXTReme
value: FetchStruct = driver.multiEval.inbandEmission.cc.margin.extreme.
↪fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return the limit line margin results for the CC<no> diagram. The CURRent margin indicates the minimum (vertical) distance between the inband emissions limit line and the current trace. A negative result indicates that the limit is exceeded. The AVERage, EXTReme and SDEViation values are calculated from the current margins. The margin results cannot be displayed at the GUI.

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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.multiEval.inbandEmission.cc.margin.extreme.clone()
```

Subgroups

6.2.8.1.1.5 RbIndex

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:CC<Nr>:MARGin:EXTReme:RBINdex
```

class RbIndexCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits. Unit: %
- Rb_Index: int: decimal Resource block index

fetch(*carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:CC<Nr>
↪:MARGin:EXTReme:RBINdex
value: FetchStruct = driver.multiEval.inbandEmission.cc.margin.extreme.rbIndex.
↪fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return resource block indices for CC<no> inband emission margins. At these RB indices, the CURRent and EXTReme margins have been detected (see method RsCmwLteMeas.MultiEval.InbandEmission.Cc.Margin.Current.fetch and . . . :EXTReme) .

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

6.2.8.1.1.6 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:CC<Nr>:MARGin:SDEViation
```

class StandardDevCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits. Unit: %
- Margin: float: float Unit: dB

fetch(*carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:CC<Nr>
↪:MARGin:SDEViation
value: FetchStruct = driver.multiEval.inbandEmission.cc.margin.standardDev.
↪fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return the limit line margin results for the CC<no> diagram. The CURRent margin indicates the minimum (vertical) distance between the inband emissions limit line and the current trace. A negative result indicates that the limit is exceeded. The AVERage, EXTReme and SDEViation values are calculated from the current margins. The margin results cannot be displayed at the GUI.

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

6.2.9 ListPy

class ListPyCls

ListPy commands group definition. 353 total commands, 9 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.9.1 Aclr

class AclrCls

Aclr commands group definition. 22 total commands, 4 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.9.1.1 Dallocation

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:DALLocation
```

class DallocationCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Nr_Res_Blocks: List[int]: decimal Number of allocated resource blocks
- Offset_Res_Blocks: List[int]: decimal Offset of the first allocated resource block from the edge of the allocated UL transmission bandwidth

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:DALLocation
value: FetchStruct = driver.multiEval.listPy.aclr.dallocation.fetch()
```

Return the detected allocation for all measured list mode segments. The result is determined from the last measured slot of the statistical length of a segment. The individual measurements provide the same result when measuring the same slot. However different statistical lengths can be defined for the measurements so that the measured slots and returned results can differ. The results are returned as pairs per segment: <Reliability>, {<NrResBlocks>, <OffsetResBlocks>}Seg 1, {<NrResBlocks>, <OffsetResBlocks>}Seg 2, ...

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

6.2.9.1.2 DchType

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:DCHType
```

class DchTypeCls

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

fetch() → List[UplinkChannelType]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:DCHType
value: List[enums.UplinkChannelType] = driver.multiEval.listPy.aclr.dchType.
↪fetch()
```

Return the uplink channel type for all measured list mode segments. The result is determined from the last measured slot of the statistical length of a segment. The individual measurements provide the same result when measuring the same slot. However different statistical lengths can be defined for the measurements so that the measured slots and returned results can differ.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
channel_type: PUSCh | PUCCh Comma-separated list of values, one per measured segment

6.2.9.1.3 Eutra

class EutraCls

Eutra 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.multiEval.listPy.aclr.eutra.clone()
```

Subgroups

6.2.9.1.3.1 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:EUTRa:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:EUTRa:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[ResultStatus2]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:EUTRa:AVERage
value: List[enums.ResultStatus2] = driver.multiEval.listPy.aclr.eutra.average.
↪calculate()
```

Return the power in the allocated E-UTRA channel for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
eutra: float Comma-separated list of values, one per measured segment Unit: dBm

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:EUTRa:AVERage
value: List[float] = driver.multiEval.listPy.aclr.eutra.average.fetch()
```

Return the power in the allocated E-UTRA channel for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
eutra: float Comma-separated list of values, one per measured segment Unit: dBm

6.2.9.1.3.2 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:EUTRa:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:EUTRa:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[ResultStatus2]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:EUTRa:CURRent
value: List[enums.ResultStatus2] = driver.multiEval.listPy.aclr.eutra.current.
↪calculate()
```

Return the power in the allocated E-UTRA channel for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
eutra: float Comma-separated list of values, one per measured segment Unit: dBm

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:EUTRa:CURRent
value: List[float] = driver.multiEval.listPy.aclr.eutra.current.fetch()
```

Return the power in the allocated E-UTRA channel for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
eutra: float Comma-separated list of values, one per measured segment Unit: dBm

6.2.9.1.3.3 Negativ

class NegativCls

Negativ 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.multiEval.listPy.aclr.eutra.negativ.clone()
```

Subgroups

6.2.9.1.3.4 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:EUTRa:NEGativ:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:EUTRa:NEGativ:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[ResultStatus2]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:ACLR:EUTRa:NEGativ:AVERage
value: List[enums.ResultStatus2] = driver.multiEval.listPy.aclr.eutra.negativ.
↪average.calculate()
```

Return the ACLR for the first adjacent E-UTRA channel above (POSitiv) or below (NEGativ) the carrier frequency for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
eutra_negativ: float Comma-separated list of values, one per measured segment Unit: dB

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:ACLR:EUTRa:NEGativ:AVERage
value: List[float] = driver.multiEval.listPy.aclr.eutra.negativ.average.fetch()
```

Return the ACLR for the first adjacent E-UTRA channel above (POSitiv) or below (NEGativ) the carrier frequency for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
eutra_negativ: float Comma-separated list of values, one per measured segment Unit: dB

6.2.9.1.3.5 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:EUTRa:NEGativ:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:EUTRa:NEGativ:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[ResultStatus2]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:ACLR:EUTRa:NEGativ:CURRent
value: List[enums.ResultStatus2] = driver.multiEval.listPy.aclr.eutra.negativ.
↪current.calculate()
```

Return the ACLR for the first adjacent E-UTRA channel above (POSitiv) or below (NEGativ) the carrier frequency for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
eutra_negativ: float Comma-separated list of values, one per measured segment Unit: dB

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:ACLR:EUTRa:NEGativ:CURRent
value: List[float] = driver.multiEval.listPy.aclr.eutra.negativ.current.fetch()
```

Return the ACLR for the first adjacent E-UTRA channel above (POSitiv) or below (NEGativ) the carrier frequency for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
eutra_negativ: float Comma-separated list of values, one per measured segment Unit: dB

6.2.9.1.3.6 Positiv

class PositivCls

Positiv 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.multiEval.listPy.aclr.eutra.positiv.clone()
```

Subgroups

6.2.9.1.3.7 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:EUTRa:POSitiv:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:EUTRa:POSitiv:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[ResultStatus2]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:ACLR:EUTRa:POSitiv:AVERage
value: List[enums.ResultStatus2] = driver.multiEval.listPy.aclr.eutra.positiv.
↪average.calculate()
```

Return the ACLR for the first adjacent E-UTRA channel above (POSitiv) or below (NEGativ) the carrier frequency for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
eutra_positiv: float Comma-separated list of values, one per measured segment Unit: dB

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:ACLR:EUTRa:POSitiv:AVERage
value: List[float] = driver.multiEval.listPy.aclr.eutra.positiv.average.fetch()
```

Return the ACLR for the first adjacent E-UTRA channel above (POSitiv) or below (NEGativ) the carrier frequency for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
eutra_positiv: float Comma-separated list of values, one per measured segment Unit: dB

6.2.9.1.3.8 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:EUTRa:POSitiv:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:EUTRa:POSitiv:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[ResultStatus2]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:ACLR:EUTRa:POSitiv:CURRent
value: List[enums.ResultStatus2] = driver.multiEval.listPy.aclr.eutra.positiv.
↪current.calculate()
```

Return the ACLR for the first adjacent E-UTRA channel above (POSitiv) or below (NEGativ) the carrier frequency for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
eutra_positiv: float Comma-separated list of values, one per measured segment Unit: dB

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:ACLR:EUTRa:POSitiv:CURRent
value: List[float] = driver.multiEval.listPy.aclr.eutra.positiv.current.fetch()
```

Return the ACLR for the first adjacent E-UTRA channel above (POSitiv) or below (NEGativ) the carrier frequency for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
eutra_positiv: float Comma-separated list of values, one per measured segment Unit: dB

6.2.9.1.4 Utra<UtraAdjChannel>

RepCap Settings

```
# Range: Ch1 .. Ch2
rc = driver.multiEval.listPy.aclr.utra.repcap_utraAdjChannel_get()
driver.multiEval.listPy.aclr.utra.repcap_utraAdjChannel_set(repcap.UtraAdjChannel.Ch1)
```

class UtraCls

Utra commands group definition. 8 total commands, 2 Subgroups, 0 group commands Repeated Capability: UtraAdjChannel, default value after init: UtraAdjChannel.Ch1

Cloning the Group

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

Subgroups

6.2.9.1.4.1 Negativ

class NegativCls

Negativ 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.multiEval.listPy.aclr.utra.negativ.clone()
```

Subgroups

6.2.9.1.4.2 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr>:NEGativ:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr>:NEGativ:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*utraAdjChannel=UtraAdjChannel.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr>
↪:NEGativ:AVERage
value: List[enums.ResultStatus2] = driver.multiEval.listPy.aclr.utra.negativ.
↪average.calculate(utraAdjChannel = repcap.UtraAdjChannel.Default)
```

Return the ACLR for the first or second adjacent UTRA channel above (POSitiv) or below (NEGativ) the carrier frequency for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param utraAdjChannel
optional repeated capability selector. Default value: Ch1 (settable in the interface 'Utra')

return
utra_negativ: No help available

fetch(*utraAdjChannel=UtraAdjChannel.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr>
↪:NEGativ:AVERage
value: List[float] = driver.multiEval.listPy.aclr.utra.negativ.average.
↪fetch(utraAdjChannel = repcap.UtraAdjChannel.Default)
```

Return the ACLR for the first or second adjacent UTRA channel above (POSitiv) or below (NEGativ) the carrier frequency for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param utraAdjChannel
optional repeated capability selector. Default value: Ch1 (settable in the interface 'Utra')

return
utra_negativ: No help available

6.2.9.1.4.3 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr>:NEGativ:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr>:NEGativ:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*utraAdjChannel=UtraAdjChannel.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr>
↪:NEGativ:CURRent
value: List[enums.ResultStatus2] = driver.multiEval.listPy.aclr.utra.negativ.
↪current.calculate(utraAdjChannel = repcap.UtraAdjChannel.Default)
```

Return the ACLR for the first or second adjacent UTRA channel above (POSitiv) or below (NEGativ) the carrier frequency for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param utraAdjChannel
optional repeated capability selector. Default value: Ch1 (settable in the interface 'Utra')

return
utra_negativ: No help available

fetch(*utraAdjChannel=UtraAdjChannel.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr>
↪:NEGativ:CURRent
value: List[float] = driver.multiEval.listPy.aclr.utra.negativ.current.
↪fetch(utraAdjChannel = repcap.UtraAdjChannel.Default)
```

Return the ACLR for the first or second adjacent UTRA channel above (POSitiv) or below (NEGativ) the carrier frequency for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param utraAdjChannel
optional repeated capability selector. Default value: Ch1 (settable in the interface 'Utra')

return
utra_negativ: No help available

6.2.9.1.4.4 Positiv

class PositivCls

Positiv 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.multiEval.listPy.aclr.utra.positiv.clone()
```

Subgroups

6.2.9.1.4.5 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr>:POSitiv:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr>:POSitiv:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*utraAdjChannel=UtraAdjChannel.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr>
↪:POSitiv:AVERage
value: List[enums.ResultStatus2] = driver.multiEval.listPy.aclr.utra.positiv.
↪average.calculate(utraAdjChannel = repcap.UtraAdjChannel.Default)
```

Return the ACLR for the first or second adjacent UTRA channel above (POSitiv) or below (NEGativ) the carrier frequency for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param utraAdjChannel
optional repeated capability selector. Default value: Ch1 (settable in the interface 'Utra')

return
utra_positiv: No help available

fetch(*utraAdjChannel=UtraAdjChannel.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr>
↪:POSitiv:AVERage
value: List[float] = driver.multiEval.listPy.aclr.utra.positiv.average.
↪fetch(utraAdjChannel = repcap.UtraAdjChannel.Default)
```

Return the ACLR for the first or second adjacent UTRA channel above (POSitiv) or below (NEGativ) the carrier frequency for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param utraAdjChannel
optional repeated capability selector. Default value: Ch1 (settable in the interface 'Utra')

return
utra_positiv: No help available

6.2.9.1.4.6 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr>:POSitiv:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr>:POSitiv:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*utraAdjChannel=UtraAdjChannel.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr>
↪:POSitiv:CURRent
value: List[enums.ResultStatus2] = driver.multiEval.listPy.aclr.utra.positiv.
↪current.calculate(utraAdjChannel = repcap.UtraAdjChannel.Default)
```

Return the ACLR for the first or second adjacent UTRA channel above (POSitiv) or below (NEGativ) the carrier frequency for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param utraAdjChannel
optional repeated capability selector. Default value: Ch1 (settable in the interface 'Utra')

return
utra_positiv: No help available

fetch(*utraAdjChannel=UtraAdjChannel.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr>
↪:POSitiv:CURRent
value: List[float] = driver.multiEval.listPy.aclr.utra.positiv.current.
↪fetch(utraAdjChannel = repcap.UtraAdjChannel.Default)
```

Return the ACLR for the first or second adjacent UTRA channel above (POSitiv) or below (NEGativ) the carrier frequency for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param utraAdjChannel
optional repeated capability selector. Default value: Ch1 (settable in the interface 'Utra')

return
utra_positiv: No help available

6.2.9.2 EsFlatness

class EsFlatnessCls

EsFlatness commands group definition. 30 total commands, 5 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.9.2.1 Difference<Difference>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.multiEval.listPy.esFlatness.difference.repcap_difference_get()
driver.multiEval.listPy.esFlatness.difference.repcap_difference_set(repcap.Difference.
↪Nr1)
```

class DifferenceCls

Difference commands group definition. 7 total commands, 4 Subgroups, 0 group commands Repeated Capability: Difference, default value after init: Difference.Nr1

Cloning the Group

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

Subgroups

6.2.9.2.1.1 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:DIFFerence<nr>:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:DIFFerence<nr>:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*difference=Difference.Default*) → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:ESFLatness:DIFFerence<nr>:AVERage
value: List[float or bool] = driver.multiEval.listPy.esFlatness.difference.
↪average.calculate(difference = repcap.Difference.Default)
```

Return equalizer spectrum flatness single value results (differences between ranges) for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param difference
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Difference')

return
difference: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: dB

fetch(*difference=Difference.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:DIFFerence
↪<nr>:AVERage
value: List[float] = driver.multiEval.listPy.esFlatness.difference.average.
↪fetch(difference = repcap.Difference.Default)
```

Return equalizer spectrum flatness single value results (differences between ranges) for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param difference
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Difference')

return
difference: float Comma-separated list of values, one per measured segment Unit: dB

6.2.9.2.1.2 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:DIFFerence<nr>:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:DIFFerence<nr>:CURRent
```

`class CurrentCls`

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

`calculate`(*difference=Difference.Default*) → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:ESFLatness:DIFFerence<nr>:CURRent
value: List[float or bool] = driver.multiEval.listPy.esFlatness.difference.
↪current.calculate(difference = repcap.Difference.Default)
```

Return equalizer spectrum flatness single value results (differences between ranges) for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param difference
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Difference')

return
difference: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: dB

`fetch`(*difference=Difference.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:DIFFerence
↪<nr>:CURRent
value: List[float] = driver.multiEval.listPy.esFlatness.difference.current.
↪fetch(difference = repcap.Difference.Default)
```

Return equalizer spectrum flatness single value results (differences between ranges) for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param difference
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Difference')

return
difference: float Comma-separated list of values, one per measured segment Unit: dB

6.2.9.2.1.3 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:DIFFerence<nr>:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:DIFFerence<nr>:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*difference=Difference.Default*) → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:ESFLatness:DIFFerence<nr>:EXTReme
value: List[float or bool] = driver.multiEval.listPy.esFlatness.difference.
↪extreme.calculate(difference = repcap.Difference.Default)
```

Return equalizer spectrum flatness single value results (differences between ranges) for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param difference
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Difference')

return
difference: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: dB

fetch(*difference=Difference.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:DIFFerence
↪<nr>:EXTReme
value: List[float] = driver.multiEval.listPy.esFlatness.difference.extreme.
↪fetch(difference = repcap.Difference.Default)
```

Return equalizer spectrum flatness single value results (differences between ranges) for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param difference
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Difference')

return
difference: float Comma-separated list of values, one per measured segment Unit: dB

6.2.9.2.1.4 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:DIFFerence<nr>:SDEViation
```

class StandardDevCls

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

fetch(*difference=Difference.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:DIFFerence
↪<nr>:SDEViation
value: List[float] = driver.multiEval.listPy.esFlatness.difference.standardDev.
↪fetch(difference = repcap.Difference.Default)
```

Return equalizer spectrum flatness single value results (differences between ranges) for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param difference
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Difference')

return
difference: float Comma-separated list of values, one per measured segment Unit: dB

6.2.9.2.2 Maxr<MaxRange>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.multiEval.listPy.esFlatness.maxr.repcap_maxRange_get()
driver.multiEval.listPy.esFlatness.maxr.repcap_maxRange_set(repcap.MaxRange.Nr1)
```

class MaxrCls

Maxr commands group definition. 7 total commands, 4 Subgroups, 0 group commands Repeated Capability: MaxRange, default value after init: MaxRange.Nr1

Cloning the Group

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

Subgroups

6.2.9.2.2.1 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MAXR<nr>:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MAXR<nr>:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*maxRange=MaxRange.Default*) → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MAXR<nr>
↪:AVERage
value: List[float or bool] = driver.multiEval.listPy.esFlatness.maxr.average.
↪calculate(maxRange = repcap.MaxRange.Default)
```

Return equalizer spectrum flatness single value results (maximum within a range) for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param maxRange
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Maxr')

return
maxr: (float or boolean items) float Comma-separated list of values, one per measured segment. Unit: dB

fetch(*maxRange=MaxRange.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MAXR<nr>
↪:AVERage
value: List[float] = driver.multiEval.listPy.esFlatness.maxr.average.
↪fetch(maxRange = repcap.MaxRange.Default)
```

Return equalizer spectrum flatness single value results (maximum within a range) for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param maxRange
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Maxr')

return
maxr: float Comma-separated list of values, one per measured segment. Unit: dB

6.2.9.2.2.2 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MAXR<nr>:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MAXR<nr>:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*maxRange=MaxRange.Default*) → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MAXR<nr>
↪:CURRent
value: List[float or bool] = driver.multiEval.listPy.esFlatness.maxr.current.
↪calculate(maxRange = repcap.MaxRange.Default)
```

Return equalizer spectrum flatness single value results (maximum within a range) for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param maxRange
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Maxr')

return
maxr: (float or boolean items) float Comma-separated list of values, one per measured segment. Unit: dB

fetch(*maxRange=MaxRange.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MAXR<nr>
↪:CURRent
value: List[float] = driver.multiEval.listPy.esFlatness.maxr.current.
↪fetch(maxRange = repcap.MaxRange.Default)
```

Return equalizer spectrum flatness single value results (maximum within a range) for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param maxRange
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Maxr')

return
maxr: float Comma-separated list of values, one per measured segment. Unit: dB

6.2.9.2.2.3 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MAXR<nr>:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MAXR<nr>:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*maxRange=MaxRange.Default*) → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MAXR<nr>
↪:EXTReme
value: List[float or bool] = driver.multiEval.listPy.esFlatness.maxr.extreme.
↪calculate(maxRange = repcap.MaxRange.Default)
```

Return equalizer spectrum flatness single value results (maximum within a range) for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param maxRange
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Maxr')

return
maxr: (float or boolean items) float Comma-separated list of values, one per measured segment. Unit: dB

fetch(*maxRange=MaxRange.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MAXR<nr>
↪:EXTReme
value: List[float] = driver.multiEval.listPy.esFlatness.maxr.extreme.
↪fetch(maxRange = repcap.MaxRange.Default)
```

Return equalizer spectrum flatness single value results (maximum within a range) for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param maxRange
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Maxr')

return
maxr: float Comma-separated list of values, one per measured segment. Unit: dB

6.2.9.2.2.4 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MAXR<nr>:SDEViation
```

class StandardDevCls

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

fetch(*maxRange=MaxRange.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MAXR<nr>
↪:SDEViation
value: List[float] = driver.multiEval.listPy.esFlatness.maxr.standardDev.
↪fetch(maxRange = repcap.MaxRange.Default)
```

Return equalizer spectrum flatness single value results (maximum within a range) for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param maxRange
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Maxr')

return
maxr: float Comma-separated list of values, one per measured segment. Unit: dB

6.2.9.2.3 Minr<MinRange>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.multiEval.listPy.esFlatness.minr.repcap_minRange_get()
driver.multiEval.listPy.esFlatness.minr.repcap_minRange_set(repcap.MinRange.Nr1)
```

class MinrCls

Minr commands group definition. 7 total commands, 4 Subgroups, 0 group commands Repeated Capability: MinRange, default value after init: MinRange.Nr1

Cloning the Group

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

Subgroups

6.2.9.2.3.1 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MINR<nr>:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MINR<nr>:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*minRange=MinRange.Default*) → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MINR<nr>
↪:AVERage
value: List[float or bool] = driver.multiEval.listPy.esFlatness.minr.average.
↪calculate(minRange = repcap.MinRange.Default)
```

Return equalizer spectrum flatness single value results (minimum within a range) for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param minRange
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Minr')

return
minr: (float or boolean items) float Comma-separated list of values, one per measured segment. Unit: dB

fetch(*minRange=MinRange.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MINR<nr>
↪:AVERage
value: List[float] = driver.multiEval.listPy.esFlatness.minr.average.
↪fetch(minRange = repcap.MinRange.Default)
```

Return equalizer spectrum flatness single value results (minimum within a range) for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param minRange
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Minr')

return
minr: float Comma-separated list of values, one per measured segment. Unit: dB

6.2.9.2.3.2 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MINR<nr>:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MINR<nr>:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*minRange=MinRange.Default*) → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MINR<nr>
↪:CURRent
value: List[float or bool] = driver.multiEval.listPy.esFlatness.minr.current.
↪calculate(minRange = repcap.MinRange.Default)
```

Return equalizer spectrum flatness single value results (minimum within a range) for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param minRange
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Minr')

return
minr: (float or boolean items) float Comma-separated list of values, one per measured segment. Unit: dB

fetch(*minRange=MinRange.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MINR<nr>
↪:CURRent
value: List[float] = driver.multiEval.listPy.esFlatness.minr.current.
↪fetch(minRange = repcap.MinRange.Default)
```

Return equalizer spectrum flatness single value results (minimum within a range) for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param minRange
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Minr')

return
minr: float Comma-separated list of values, one per measured segment. Unit: dB

6.2.9.2.3.3 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MINR<nr>:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MINR<nr>:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*minRange=MinRange.Default*) → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MINR<nr>
↪:EXTReme
value: List[float or bool] = driver.multiEval.listPy.esFlatness.minr.extreme.
↪calculate(minRange = repcap.MinRange.Default)
```

Return equalizer spectrum flatness single value results (minimum within a range) for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param minRange
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Minr')

return
minr: (float or boolean items) float Comma-separated list of values, one per measured segment. Unit: dB

fetch(*minRange=MinRange.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MINR<nr>
↪:EXTReme
value: List[float] = driver.multiEval.listPy.esFlatness.minr.extreme.
↪fetch(minRange = repcap.MinRange.Default)
```

Return equalizer spectrum flatness single value results (minimum within a range) for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param minRange
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Minr')

return
minr: float Comma-separated list of values, one per measured segment. Unit: dB

6.2.9.2.3.4 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MINR<nr>:SDEViation
```

class StandardDevCls

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

fetch(*minRange=MinRange.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MINR<nr>
↪:SDEViation
value: List[float] = driver.multiEval.listPy.esFlatness.minr.standardDev.
↪fetch(minRange = repcap.MinRange.Default)
```

Return equalizer spectrum flatness single value results (minimum within a range) for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param minRange
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Minr')

return
minr: float Comma-separated list of values, one per measured segment. Unit: dB

6.2.9.2.4 Ripple<Ripple>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.multiEval.listPy.esFlatness.ripple.repcap_ripple_get()
driver.multiEval.listPy.esFlatness.ripple.repcap_ripple_set(repcap.Ripple.Nr1)
```

class RippleCls

Ripple commands group definition. 7 total commands, 4 Subgroups, 0 group commands Repeated Capability: Ripple, default value after init: Ripple.Nr1

Cloning the Group

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

Subgroups

6.2.9.2.4.1 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:RIPPle<nr>:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:RIPPle<nr>:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*ripple=Ripple.Default*) → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:RIPPle
↪<nr>:AVERage
value: List[float or bool] = driver.multiEval.listPy.esFlatness.ripple.average.
↪calculate(ripple = repcap.Ripple.Default)
```

Return equalizer spectrum flatness single value results (ripple 1 or ripple 2) for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param ripple
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Ripple')

return
ripple: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: dB

fetch(*ripple=Ripple.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:RIPPle<nr>
↪:AVERage
value: List[float] = driver.multiEval.listPy.esFlatness.ripple.average.
↪fetch(ripple = repcap.Ripple.Default)
```

Return equalizer spectrum flatness single value results (ripple 1 or ripple 2) for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param ripple
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Ripple')

return
ripple: float Comma-separated list of values, one per measured segment Unit: dB

6.2.9.2.4.2 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:RIPPle<nr>:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:RIPPle<nr>:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*ripple=Ripple.Default*) → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:RIPPle
↪<nr>:CURRent
value: List[float or bool] = driver.multiEval.listPy.esFlatness.ripple.current.
↪calculate(ripple = repcap.Ripple.Default)
```

Return equalizer spectrum flatness single value results (ripple 1 or ripple 2) for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param ripple
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Ripple')

return
ripple: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: dB

fetch(*ripple=Ripple.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:RIPPle<nr>
↪:CURRent
value: List[float] = driver.multiEval.listPy.esFlatness.ripple.current.
↪fetch(ripple = repcap.Ripple.Default)
```

Return equalizer spectrum flatness single value results (ripple 1 or ripple 2) for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param ripple
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Ripple')

return
ripple: float Comma-separated list of values, one per measured segment Unit: dB

6.2.9.2.4.3 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:RIPPle<nr>:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:RIPPle<nr>:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*ripple=Ripple.Default*) → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:RIPPle
↪<nr>:EXTReme
value: List[float or bool] = driver.multiEval.listPy.esFlatness.ripple.extreme.
↪calculate(ripple = repcap.Ripple.Default)
```

Return equalizer spectrum flatness single value results (ripple 1 or ripple 2) for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param ripple
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Ripple')

return
ripple: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: dB

fetch(*ripple=Ripple.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:RIPPle<nr>
↪:EXTReme
value: List[float] = driver.multiEval.listPy.esFlatness.ripple.extreme.
↪fetch(ripple = repcap.Ripple.Default)
```

Return equalizer spectrum flatness single value results (ripple 1 or ripple 2) for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param ripple
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Ripple')

return
ripple: float Comma-separated list of values, one per measured segment Unit: dB

6.2.9.2.4.4 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:RIPPle<nr>:SDEViation
```

class StandardDevCls

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

fetch(*ripple=Ripple.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:RIPPle<nr>
↪:SDEViation
value: List[float] = driver.multiEval.listPy.esFlatness.ripple.standardDev.
↪fetch(ripple = repcap.Ripple.Default)
```

Return equalizer spectrum flatness single value results (ripple 1 or ripple 2) for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param ripple
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Ripple')

return
ripple: float Comma-separated list of values, one per measured segment Unit: dB

6.2.9.2.5 ScIndex

class ScIndexCls

ScIndex 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.multiEval.listPy.esFlatness.scIndex.clone()
```

Subgroups

6.2.9.2.5.1 Maximum<MaxRange>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.multiEval.listPy.esFlatness.scIndex.maximum.repcap_maxRange_get()
driver.multiEval.listPy.esFlatness.scIndex.maximum.repcap_maxRange_set(repcap.MaxRange.
↪Nr1)
```

class MaximumCls

Maximum commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability: MaxRange, default value after init: MaxRange.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.esFlatness.scIndex.maximum.clone()
```

Subgroups

6.2.9.2.5.2 Current

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:SCINdex:MAXimum<nr>:CURRent
```

class CurrentCls

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

fetch(*maxRange=MaxRange.Default*) → List[int]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:ESFLatness:SCINdex:MAXimum<nr>:CURRent
value: List[int] = driver.multiEval.listPy.esFlatness.scIndex.maximum.current.
↪fetch(maxRange = repcap.MaxRange.Default)
```

Return subcarrier indices of the equalizer spectrum flatness measurement for all measured list mode segments. At these SC indices, the current MINimum or MAXimum power of the equalizer coefficients has been detected within the selected range.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param maxRange
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Maximum')

return
maximum: No help available

6.2.9.2.5.3 Minimum<MinRange>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.multiEval.listPy.esFlatness.scIndex.minimum.repcap_minRange_get()
driver.multiEval.listPy.esFlatness.scIndex.minimum.repcap_minRange_set(repcap.MinRange.
↪Nr1)
```

class MinimumCls

Minimum commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability: MinRange, default value after init: MinRange.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.esFlatness.scIndex.minimum.clone()
```

Subgroups

6.2.9.2.5.4 Current

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:SCINdex:MINimum<nr>:CURRent
```

class CurrentCls

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

fetch(*minRange=MinRange.Default*) → List[int]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:ESFLatness:SCINdex:MINimum<nr>:CURRent
value: List[int] = driver.multiEval.listPy.esFlatness.scIndex.minimum.current.
↪fetch(minRange = repcap.MinRange.Default)
```

Return subcarrier indices of the equalizer spectrum flatness measurement for all measured list mode segments. At these SC indices, the current MINimum or MAXimum power of the equalizer coefficients has been detected within the selected range.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param minRange
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Minimum')

return
minimum: No help available

6.2.9.3 InbandEmission

class InbandEmissionCls

InbandEmission 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.multiEval.listPy.inbandEmission.clone()
```

Subgroups

6.2.9.3.1 Margin

class MarginCls

Margin commands group definition. 6 total commands, 5 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.9.3.1.1 Average

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:IEMission:MARGin:AVERage
```

class AverageCls

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

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:IEMission:MARGin:AVERage
value: List[float] = driver.multiEval.listPy.inbandEmission.margin.average.
↪fetch()
```

Return the inband emission limit line margin results for all measured list mode segments. The CURRent margins indicate the minimum (vertical) distance between the limit line and the current trace. A negative result indicates that the limit is exceeded. The AVERage, EXTReme and SDEViation values are calculated from the current margins.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
margin: float Comma-separated list of values, one per measured segment Unit: dB

6.2.9.3.1.2 Current

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:IEMission:MARGin:CURRent
```

class CurrentCls

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

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:IEMission:MARGin:CURRent
value: List[float] = driver.multiEval.listPy.inbandEmission.margin.current.
↪fetch()
```

Return the inband emission limit line margin results for all measured list mode segments. The CURRent margins indicate the minimum (vertical) distance between the limit line and the current trace. A negative result indicates that the limit is exceeded. The AVERage, EXTReme and SDEViation values are calculated from the current margins.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
margin: float Comma-separated list of values, one per measured segment Unit: dB

6.2.9.3.1.3 Extreme

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:IEMission:MARGin:EXTReme
```

class ExtremeCls

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

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:IEMission:MARGin:EXTReme
value: List[float] = driver.multiEval.listPy.inbandEmission.margin.extreme.
↪fetch()
```

Return the inband emission limit line margin results for all measured list mode segments. The CURRent margins indicate the minimum (vertical) distance between the limit line and the current trace. A negative result indicates that the limit is exceeded. The AVERage, EXTReme and SDEViation values are calculated from the current margins.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
margin: float Comma-separated list of values, one per measured segment Unit: dB

6.2.9.3.1.4 RbIndex

class RbIndexCls

RbIndex 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.multiEval.listPy.inbandEmission.margin.rbIndex.clone()
```

Subgroups

6.2.9.3.1.5 Current

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:IEMission:MARGin:RBINdex:CURRent
```

class CurrentCls

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

fetch() → List[int]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:IEMission:MARGin:RBINdex:CURRent
value: List[int] = driver.multiEval.listPy.inbandEmission.margin.rbIndex.
↪current.fetch()
```

Return resource block indices of the inband emission measurement for all measured list mode segments. At these RB indices, the CURRent and EXTReme margins have been detected.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return

rb_index: decimal Comma-separated list of values, one per measured segment

6.2.9.3.1.6 Extreme

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:IEMission:MARGin:RBINdex:EXTReme
```

class ExtremeCls

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

fetch() → List[int]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:IEMission:MARGin:RBINdex:EXTReme
value: List[int] = driver.multiEval.listPy.inbandEmission.margin.rbIndex.
↪extreme.fetch()
```

Return resource block indices of the inband emission measurement for all measured list mode segments. At these RB indices, the CURRent and EXTReme margins have been detected.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
rb_index: decimal Comma-separated list of values, one per measured segment

6.2.9.3.1.7 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:IEMission:MARGin:SDEViation
```

class StandardDevCls

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

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:IEMission:MARGin:SDEViation
value: List[float] = driver.multiEval.listPy.inbandEmission.margin.standardDev.
↪fetch()
```

Return the inband emission limit line margin results for all measured list mode segments. The CURRent margins indicate the minimum (vertical) distance between the limit line and the current trace. A negative result indicates that the limit is exceeded. The AVERage, EXTReme and SDEViation values are calculated from the current margins.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
margin: float Comma-separated list of values, one per measured segment Unit: dB

6.2.9.4 Modulation

class ModulationCls

Modulation commands group definition. 178 total commands, 13 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.9.4.1 Dallocation

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:DALLocation
```

class DallocationCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Nr_Res_Blocks: List[int]: decimal Number of allocated resource blocks
- Offset_Res_Blocks: List[int]: decimal Offset of the first allocated resource block from the edge of the allocated UL transmission bandwidth

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:DALLocation
value: FetchStruct = driver.multiEval.listPy.modulation.dallocation.fetch()
```

Return the detected allocation for all measured list mode segments. The result is determined from the last measured slot of the statistical length of a segment. The individual measurements provide the same result when measuring the same slot. However different statistical lengths can be defined for the measurements so that the measured slots and returned results can differ. The results are returned as pairs per segment: <Reliability>, {<NrResBlocks>, <OffsetResBlocks>}Seg 1, {<NrResBlocks>, <OffsetResBlocks>}Seg 2, ...

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

6.2.9.4.2 DchType

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:DCHType
```

class DchTypeCls

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

fetch() → List[UplinkChannelType]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:DCHType
value: List[enums.UplinkChannelType] = driver.multiEval.listPy.modulation.
↪dchType.fetch()
```

Return the uplink channel type for all measured list mode segments. The result is determined from the last measured slot of the statistical length of a segment. The individual measurements provide the same result when measuring the same slot. However different statistical lengths can be defined for the measurements so that the measured slots and returned results can differ.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
channel_type: PUSCh | PUCCh Comma-separated list of values, one per measured segment

6.2.9.4.3 Dmodulation

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:DMODulation
```

class DmodulationCls

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

fetch() → List[Modulation]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:DMODulation
value: List[enums.Modulation] = driver.multiEval.listPy.modulation.dmodulation.
↪fetch()
```

Return the detected modulation scheme for all measured list mode segments. The result is determined from the last measured slot of the statistical length of a segment. If channel type PUCCH is detected, QPSK is returned as modulation type because the QPSK limits are applied in that case.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
modulation: QPSK | Q16 | Q64 | Q256 Comma-separated list of values, one per measured segment QPSK, 16-QAM, 64-QAM, 256-QAM

6.2.9.4.4 Evm

class EvmCls

Evm commands group definition. 42 total commands, 3 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.9.4.4.1 Dmrs

class DmrsCls

Dmrs commands group definition. 14 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.evm.dmrs.clone()
```

Subgroups

6.2.9.4.4.2 High

class HighCls

High commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.evm.dmrs.high.clone()
```

Subgroups

6.2.9.4.4.3 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:DMRS:HIGH:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:DMRS:HIGH:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:DMRS:HIGH:AVERage
value: List[float or bool] = driver.multiEval.listPy.modulation.evm.dmrs.high.
↪average.calculate()
```

Return error vector magnitude DMRS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_dmrs_high: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: %

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:DMRS:HIGH:AVERage
value: List[float] = driver.multiEval.listPy.modulation.evm.dmrs.high.average.
↪fetch()
```

Return error vector magnitude DMRS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_dmrs_high: float Comma-separated list of values, one per measured segment Unit: %

6.2.9.4.4.4 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:DMRS:HIGH:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:DMRS:HIGH:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:DMRS:HIGH:CURRent
value: List[float or bool] = driver.multiEval.listPy.modulation.evm.dmrs.high.
↪current.calculate()
```

Return error vector magnitude DMRS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_dmrs_high: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: %

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:DMRS:HIGH:CURRent
value: List[float] = driver.multiEval.listPy.modulation.evm.dmrs.high.current.
↪fetch()
```

Return error vector magnitude DMRS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_dmrs_high: float Comma-separated list of values, one per measured segment Unit: %

6.2.9.4.4.5 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:DMRS:HIGH:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:DMRS:HIGH:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:DMRS:HIGH:EXTReme
value: List[float or bool] = driver.multiEval.listPy.modulation.evm.dmrs.high.
↪extreme.calculate()
```

Return error vector magnitude DMRS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_dmrs_high: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: %

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:DMRS:HIGH:EXTReme
value: List[float] = driver.multiEval.listPy.modulation.evm.dmrs.high.extreme.
↪fetch()
```

Return error vector magnitude DMRS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_dmrs_high: float Comma-separated list of values, one per measured segment Unit: %

6.2.9.4.4.6 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:DMRS:HIGH:SDEViation
```

class StandardDevCls

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

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:DMRS:HIGH:SDEViation
value: List[float] = driver.multiEval.listPy.modulation.evm.dmrs.high.
↪standardDev.fetch()
```

Return error vector magnitude DMRS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return

evm_dmrs_high: float Comma-separated list of values, one per measured segment
Unit: %

6.2.9.4.4.7 Low

class LowCls

Low commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.evm.dmrs.low.clone()
```

Subgroups

6.2.9.4.4.8 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:DMRS:LOW:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:DMRS:LOW:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:DMRS:LOW:AVERage
value: List[float or bool] = driver.multiEval.listPy.modulation.evm.dmrs.low.
↪average.calculate()
```

Return error vector magnitude DMRS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_dmrs_low: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: %

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:DMRS:LOW:AVERage
value: List[float] = driver.multiEval.listPy.modulation.evm.dmrs.low.average.
↪fetch()
```

Return error vector magnitude DMRS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_dmrs_low: float Comma-separated list of values, one per measured segment Unit: %

6.2.9.4.4.9 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:DMRS:LOW:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:DMRS:LOW:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:DMRS:LOW:CURRent
value: List[float or bool] = driver.multiEval.listPy.modulation.evm.dmrs.low.
↪current.calculate()
```

Return error vector magnitude DMRS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_dmrs_low: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: %

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:DMRS:LOW:CURRent
value: List[float] = driver.multiEval.listPy.modulation.evm.dmrs.low.current.
↪fetch()
```

Return error vector magnitude DMRS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_dmrs_low: float Comma-separated list of values, one per measured segment Unit: %

6.2.9.4.4.10 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:DMRS:LOW:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:DMRS:LOW:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:DMRS:LOW:EXTReme
value: List[float or bool] = driver.multiEval.listPy.modulation.evm.dmrs.low.
↪extreme.calculate()
```

Return error vector magnitude DMRS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_dmrs_low: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: %

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:DMRS:LOW:EXTReme
value: List[float] = driver.multiEval.listPy.modulation.evm.dmrs.low.extreme.
↪fetch()
```

Return error vector magnitude DMRS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_dmrs_low: float Comma-separated list of values, one per measured segment Unit: %

6.2.9.4.4.11 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:DMRS:LOW:SDEViation
```

class StandardDevCls

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

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:DMRS:LOW:SDEViation
value: List[float] = driver.multiEval.listPy.modulation.evm.dmrs.low.
↪standardDev.fetch()
```

Return error vector magnitude DMRS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_dmrs_low: float Comma-separated list of values, one per measured segment Unit: %

6.2.9.4.4.12 Peak

class PeakCls

Peak commands group definition. 14 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.evm.peak.clone()
```

Subgroups

6.2.9.4.4.13 High

class HighCls

High commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.evm.peak.high.clone()
```

Subgroups

6.2.9.4.4.14 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:PEAK:HIGH:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:PEAK:HIGH:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:PEAK:HIGH:AVERage
value: List[float or bool] = driver.multiEval.listPy.modulation.evm.peak.high.
↪average.calculate()
```

Return error vector magnitude peak values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_peak_high: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: %

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:PEAK:HIGH:AVERage
value: List[float] = driver.multiEval.listPy.modulation.evm.peak.high.average.
↪fetch()
```

Return error vector magnitude peak values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_peak_high: float Comma-separated list of values, one per measured segment Unit: %

6.2.9.4.4.15 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:PEAK:HIGH:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:PEAK:HIGH:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:PEAK:HIGH:CURRent
value: List[float or bool] = driver.multiEval.listPy.modulation.evm.peak.high.
↪current.calculate()
```

Return error vector magnitude peak values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_peak_high: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: %

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:PEAK:HIGH:CURRent
value: List[float] = driver.multiEval.listPy.modulation.evm.peak.high.current.
↪fetch()
```

Return error vector magnitude peak values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_peak_high: float Comma-separated list of values, one per measured segment Unit: %

6.2.9.4.4.16 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:PEAK:HIGH:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:PEAK:HIGH:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:PEAK:HIGH:EXTReme
value: List[float or bool] = driver.multiEval.listPy.modulation.evm.peak.high.
↪extreme.calculate()
```

Return error vector magnitude peak values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_peak_high: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: %

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:PEAK:HIGH:EXTReme
value: List[float] = driver.multiEval.listPy.modulation.evm.peak.high.extreme.
↪fetch()
```

Return error vector magnitude peak values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_peak_high: float Comma-separated list of values, one per measured segment Unit: %

6.2.9.4.4.17 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:PEAK:HIGH:SDEViation
```

class StandardDevCls

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

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:PEAK:HIGH:SDEViation
value: List[float] = driver.multiEval.listPy.modulation.evm.peak.high.
↪standardDev.fetch()
```

Return error vector magnitude peak values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_peak_high: float Comma-separated list of values, one per measured segment
Unit: %

6.2.9.4.4.18 Low

class LowCls

Low commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.evm.peak.low.clone()
```

Subgroups

6.2.9.4.4.19 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:PEAK:LOW:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:PEAK:LOW:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:PEAK:LOW:AVERage
value: List[float or bool] = driver.multiEval.listPy.modulation.evm.peak.low.
↪average.calculate()
```

Return error vector magnitude peak values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_peak_low: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: %

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:PEAK:LOW:AVERage
value: List[float] = driver.multiEval.listPy.modulation.evm.peak.low.average.
↪fetch()
```

Return error vector magnitude peak values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_peak_low: float Comma-separated list of values, one per measured segment Unit: %

6.2.9.4.4.20 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:PEAK:LOW:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:PEAK:LOW:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:PEAK:LOW:CURRent
value: List[float or bool] = driver.multiEval.listPy.modulation.evm.peak.low.
↪current.calculate()
```

Return error vector magnitude peak values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_peak_low: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: %

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:PEAK:LOW:CURRent
value: List[float] = driver.multiEval.listPy.modulation.evm.peak.low.current.
↪fetch()
```

Return error vector magnitude peak values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_peak_low: float Comma-separated list of values, one per measured segment Unit: %

6.2.9.4.4.21 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:PEAK:LOW:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:PEAK:LOW:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:PEAK:LOW:EXTReme
value: List[float or bool] = driver.multiEval.listPy.modulation.evm.peak.low.
↪extreme.calculate()
```

Return error vector magnitude peak values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_peak_low: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: %

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:PEAK:LOW:EXTReme
value: List[float] = driver.multiEval.listPy.modulation.evm.peak.low.extreme.
↪fetch()
```

Return error vector magnitude peak values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_peak_low: float Comma-separated list of values, one per measured segment Unit: %

6.2.9.4.4.22 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:PEAK:LOW:SDEViation
```

class StandardDevCls

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

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:PEAK:LOW:SDEViation
value: List[float] = driver.multiEval.listPy.modulation.evm.peak.low.
↪standardDev.fetch()
```

Return error vector magnitude peak values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_peak_low: float Comma-separated list of values, one per measured segment Unit: %

6.2.9.4.4.23 Rms

class RmsCls

Rms commands group definition. 14 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.evm.rms.clone()
```

Subgroups

6.2.9.4.4.24 High

class HighCls

High commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.evm.rms.high.clone()
```

Subgroups

6.2.9.4.4.25 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:RMS:HIGH:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:RMS:HIGH:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:RMS:HIGH:AVERage
value: List[float or bool] = driver.multiEval.listPy.modulation.evm.rms.high.
↪average.calculate()
```

Return error vector magnitude RMS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_rms_high: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: %

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:RMS:HIGH:AVERage
value: List[float] = driver.multiEval.listPy.modulation.evm.rms.high.average.
↪fetch()
```

Return error vector magnitude RMS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_rms_high: float Comma-separated list of values, one per measured segment Unit: %

6.2.9.4.4.26 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:RMS:HIGH:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:RMS:HIGH:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:RMS:HIGH:CURRent
value: List[float or bool] = driver.multiEval.listPy.modulation.evm.rms.high.
↪current.calculate()
```

Return error vector magnitude RMS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_rms_high: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: %

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:RMS:HIGH:CURRent
value: List[float] = driver.multiEval.listPy.modulation.evm.rms.high.current.
↪fetch()
```

Return error vector magnitude RMS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_rms_high: float Comma-separated list of values, one per measured segment Unit: %

6.2.9.4.4.27 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:RMS:HIGH:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:RMS:HIGH:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:RMS:HIGH:EXTReme
value: List[float or bool] = driver.multiEval.listPy.modulation.evm.rms.high.
↪extreme.calculate()
```

Return error vector magnitude RMS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_rms_high: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: %

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:RMS:HIGH:EXTReme
value: List[float] = driver.multiEval.listPy.modulation.evm.rms.high.extreme.
↪fetch()
```

Return error vector magnitude RMS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_rms_high: float Comma-separated list of values, one per measured segment Unit: %

6.2.9.4.4.28 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:RMS:HIGH:SDEViation
```

class StandardDevCls

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

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:RMS:HIGH:SDEViation
value: List[float] = driver.multiEval.listPy.modulation.evm.rms.high.
↪standardDev.fetch()
```

Return error vector magnitude RMS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_rms_high: float Comma-separated list of values, one per measured segment Unit: %

6.2.9.4.4.29 Low

class LowCls

Low commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.evm.rms.low.clone()
```

Subgroups

6.2.9.4.4.30 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:RMS:LOW:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:RMS:LOW:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:RMS:LOW:AVERage
value: List[float or bool] = driver.multiEval.listPy.modulation.evm.rms.low.
↪average.calculate()
```

Return error vector magnitude RMS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_rms_low: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: %

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:RMS:LOW:AVERage
value: List[float] = driver.multiEval.listPy.modulation.evm.rms.low.average.
↪fetch()
```

Return error vector magnitude RMS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_rms_low: float Comma-separated list of values, one per measured segment Unit: %

6.2.9.4.4.31 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:RMS:LOW:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:RMS:LOW:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:RMS:LOW:CURRent
value: List[float or bool] = driver.multiEval.listPy.modulation.evm.rms.low.
↪current.calculate()
```

Return error vector magnitude RMS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_rms_low: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: %

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:RMS:LOW:CURRent
value: List[float] = driver.multiEval.listPy.modulation.evm.rms.low.current.
↪fetch()
```

Return error vector magnitude RMS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_rms_low: float Comma-separated list of values, one per measured segment Unit: %

6.2.9.4.4.32 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:RMS:LOW:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:RMS:LOW:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:RMS:LOW:EXTReme
value: List[float or bool] = driver.multiEval.listPy.modulation.evm.rms.low.
↪extreme.calculate()
```

Return error vector magnitude RMS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_rms_low: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: %

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:RMS:LOW:EXTReme
value: List[float] = driver.multiEval.listPy.modulation.evm.rms.low.extreme.
↪fetch()
```

Return error vector magnitude RMS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_rms_low: float Comma-separated list of values, one per measured segment Unit: %

6.2.9.4.4.33 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:RMS:LOW:SDEViation
```

class StandardDevCls

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

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:RMS:LOW:SDEViation
value: List[float] = driver.multiEval.listPy.modulation.evm.rms.low.standardDev.
↪fetch()
```

Return error vector magnitude RMS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
evm_rms_low: float Comma-separated list of values, one per measured segment Unit: %

6.2.9.4.5 FreqError

class FreqErrorCls

FreqError commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.9.4.5.1 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:FERRor:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:FERRor:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:FERRor:AVERage
value: List[float or bool] = driver.multiEval.listPy.modulation.freqError.
↪average.calculate()
```

Return carrier frequency error values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
frequency_error: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: Hz

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:FERRor:AVERage
value: List[float] = driver.multiEval.listPy.modulation.freqError.average.
↪fetch()
```

Return carrier frequency error values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
frequency_error: float Comma-separated list of values, one per measured segment Unit: Hz

6.2.9.4.5.2 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:FERRor:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:FERRor:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:FERRor:CURRent
value: List[float or bool] = driver.multiEval.listPy.modulation.freqError.
↪current.calculate()
```

Return carrier frequency error values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
frequency_error: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: Hz

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:FERRor:CURRent
value: List[float] = driver.multiEval.listPy.modulation.freqError.current.
↪fetch()
```

Return carrier frequency error values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
frequency_error: float Comma-separated list of values, one per measured segment Unit: Hz

6.2.9.4.5.3 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:FERRor:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:FERRor:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:FERRor:EXTReme
value: List[float or bool] = driver.multiEval.listPy.modulation.freqError.
↪extreme.calculate()
```

Return carrier frequency error values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
frequency_error: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: Hz

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:FERRor:EXTReme
value: List[float] = driver.multiEval.listPy.modulation.freqError.extreme.
↪fetch()
```

Return carrier frequency error values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
frequency_error: float Comma-separated list of values, one per measured segment Unit: Hz

6.2.9.4.5.4 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:FERRor:SDEViation
```

class StandardDevCls

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

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:FERRor:SDEViation
value: List[float] = driver.multiEval.listPy.modulation.freqError.standardDev.
↪fetch()
```

Return carrier frequency error values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
frequency_error: float Comma-separated list of values, one per measured segment
Unit: Hz

6.2.9.4.6 IqOffset

class IqOffsetCls

IqOffset commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.9.4.6.1 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:IQOFfset:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:IQOFfset:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:IQOFfset:AVERage
value: List[float or bool] = driver.multiEval.listPy.modulation.iqOffset.
↪average.calculate()
```

Return I/Q origin offset values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
iq_offset: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: dBc

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:IQOFfset:AVERage
value: List[float] = driver.multiEval.listPy.modulation.iqOffset.average.fetch()
```

Return I/Q origin offset values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
iq_offset: float Comma-separated list of values, one per measured segment Unit: dBc

6.2.9.4.6.2 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:IQOFfset:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:IQOFfset:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:IQOFfset:CURRent
value: List[float or bool] = driver.multiEval.listPy.modulation.iqOffset.
↪current.calculate()
```

Return I/Q origin offset values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
iq_offset: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: dBc

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:IQOFfset:CURRent
value: List[float] = driver.multiEval.listPy.modulation.iqOffset.current.fetch()
```

Return I/Q origin offset values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
iq_offset: float Comma-separated list of values, one per measured segment Unit: dBc

6.2.9.4.6.3 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:IQOFfset:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:IQOFfset:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:IQOFfset:EXTReme
value: List[float or bool] = driver.multiEval.listPy.modulation.iqOffset.
↪extreme.calculate()
```

Return I/Q origin offset values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
iq_offset: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: dBc

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:IQOFfset:EXTReme
value: List[float] = driver.multiEval.listPy.modulation.iqOffset.extreme.fetch()
```

Return I/Q origin offset values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
iq_offset: float Comma-separated list of values, one per measured segment Unit: dBc

6.2.9.4.6.4 StandardDev

SCPI Command :

```
FETCh:LTE: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:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:IQOFfset:SDEViation
value: List[float] = driver.multiEval.listPy.modulation.iqOffset.standardDev.
↪fetch()
```

Return I/Q origin offset values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
iq_offset: float Comma-separated list of values, one per measured segment Unit: dBc

6.2.9.4.7 Merror

class MerrorCls

Merror commands group definition. 42 total commands, 3 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.9.4.7.1 Dmrs

class DmrsCls

Dmrs commands group definition. 14 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.merror.dmrs.clone()
```

Subgroups

6.2.9.4.7.2 High

class HighCls

High commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.merror.dmrs.high.clone()
```

Subgroups

6.2.9.4.7.3 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:DMRS:HIGH:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:DMRS:HIGH:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:DMRS:HIGH:AVERage
value: List[float or bool] = driver.multiEval.listPy.modulation.merror.dmrs.
↪high.average.calculate()
```

Return magnitude error DMRS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
mag_err_dmrs_high: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: %

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:DMRS:HIGH:AVERage
value: List[float] = driver.multiEval.listPy.modulation.merror.dmrs.high.
↪average.fetch()
```

Return magnitude error DMRS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
mag_err_dmrs_high: float Comma-separated list of values, one per measured segment
Unit: %

6.2.9.4.7.4 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:DMRS:HIGH:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:DMRS:HIGH:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:DMRS:HIGH:CURRent
value: List[float or bool] = driver.multiEval.listPy.modulation.merror.dmrs.
↪high.current.calculate()
```

Return magnitude error DMRS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
mag_err_dmrs_high: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: %

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:DMRS:HIGH:CURRent
value: List[float] = driver.multiEval.listPy.modulation.merror.dmrs.high.
↪current.fetch()
```

Return magnitude error DMRS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
mag_err_dmrs_high: float Comma-separated list of values, one per measured segment
Unit: %

6.2.9.4.7.5 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:DMRS:HIGH:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:DMRS:HIGH:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:DMRS:HIGH:EXTReme
value: List[float or bool] = driver.multiEval.listPy.modulation.merror.dmrs.
↪high.extreme.calculate()
```

Return magnitude error DMRS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
mag_err_dmrs_high: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: %

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:DMRS:HIGH:EXTReme
value: List[float] = driver.multiEval.listPy.modulation.merror.dmrs.high.
↪extreme.fetch()
```

Return magnitude error DMRS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
mag_err_dmrs_high: float Comma-separated list of values, one per measured segment
Unit: %

6.2.9.4.7.6 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:DMRS:HIGH:SDEViation
```

class StandardDevCls

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

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:DMRS:HIGH:SDEViation
value: List[float] = driver.multiEval.listPy.modulation.merror.dmrs.high.
↪standardDev.fetch()
```

Return magnitude error DMRS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
mag_err_dmrs_high: float Comma-separated list of values, one per measured segment
Unit: %

6.2.9.4.7.7 Low

class LowCls

Low commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.merror.dmrs.low.clone()
```

Subgroups

6.2.9.4.7.8 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:DMRS:LOW:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:DMRS:LOW:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:DMRS:LOW:AVERage
value: List[float or bool] = driver.multiEval.listPy.modulation.merror.dmrs.low.
↪average.calculate()
```

Return magnitude error DMRS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
mag_err_dmrs_low: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: %

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:DMRS:LOW:AVERage
value: List[float] = driver.multiEval.listPy.modulation.merror.dmrs.low.average.
↪fetch()
```

Return magnitude error DMRS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
mag_err_dmrs_low: float Comma-separated list of values, one per measured segment Unit: %

6.2.9.4.7.9 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:DMRS:LOW:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:DMRS:LOW:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:DMRS:LOW:CURRent
value: List[float or bool] = driver.multiEval.listPy.modulation.merror.dmrs.low.
↪current.calculate()
```

Return magnitude error DMRS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
mag_err_dmrs_low: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: %

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:DMRS:LOW:CURRent
value: List[float] = driver.multiEval.listPy.modulation.merror.dmrs.low.current.
↪fetch()
```

Return magnitude error DMRS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
mag_err_dmrs_low: float Comma-separated list of values, one per measured segment Unit: %

6.2.9.4.7.10 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:DMRS:LOW:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:DMRS:LOW:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:DMRS:LOW:EXTReme
value: List[float or bool] = driver.multiEval.listPy.modulation.merror.dmrs.low.
↪extreme.calculate()
```

Return magnitude error DMRS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
mag_err_dmrs_low: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: %

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:DMRS:LOW:EXTReme
value: List[float] = driver.multiEval.listPy.modulation.merror.dmrs.low.extreme.
↪fetch()
```

Return magnitude error DMRS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
mag_err_dmrs_low: float Comma-separated list of values, one per measured segment
Unit: %

6.2.9.4.7.11 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:DMRS:LOW:SDEViation
```

class StandardDevCls

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

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:DMRS:LOW:SDEViation
value: List[float] = driver.multiEval.listPy.modulation.merror.dmrs.low.
↪standardDev.fetch()
```

Return magnitude error DMRS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
mag_err_dmrs_low: float Comma-separated list of values, one per measured segment
Unit: %

6.2.9.4.7.12 Peak

class PeakCls

Peak commands group definition. 14 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.merror.peak.clone()
```

Subgroups

6.2.9.4.7.13 High

class HighCls

High commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.merror.peak.high.clone()
```

Subgroups

6.2.9.4.7.14 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:PEAK:HIGH:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:PEAK:HIGH:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:PEAK:HIGH:AVERage
value: List[float or bool] = driver.multiEval.listPy.modulation.merror.peak.
↪high.average.calculate()
```

Return magnitude error peak values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
mag_err_peak_high: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: %

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:PEAK:HIGH:AVERage
value: List[float] = driver.multiEval.listPy.modulation.merror.peak.high.
↪average.fetch()
```

Return magnitude error peak values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
mag_err_peak_high: float Comma-separated list of values, one per measured segment
Unit: %

6.2.9.4.7.15 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:PEAK:HIGH:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:PEAK:HIGH:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:PEAK:HIGH:CURRent
value: List[float or bool] = driver.multiEval.listPy.modulation.merror.peak.
↪high.current.calculate()
```

Return magnitude error peak values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
mag_err_peak_high: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: %

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:PEAK:HIGH:CURRent
value: List[float] = driver.multiEval.listPy.modulation.merror.peak.high.
↪current.fetch()
```

Return magnitude error peak values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
mag_err_peak_high: float Comma-separated list of values, one per measured segment
Unit: %

6.2.9.4.7.16 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:PEAK:HIGH:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:PEAK:HIGH:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:PEAK:HIGH:EXTReme
value: List[float or bool] = driver.multiEval.listPy.modulation.merror.peak.
↪high.extreme.calculate()
```

Return magnitude error peak values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
mag_err_peak_high: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: %

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:PEAK:HIGH:EXTReme
value: List[float] = driver.multiEval.listPy.modulation.merror.peak.high.
↪extreme.fetch()
```

Return magnitude error peak values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
mag_err_peak_high: float Comma-separated list of values, one per measured segment Unit: %

6.2.9.4.7.17 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:PEAK:HIGH:SDEViation
```

class StandardDevCls

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

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:PEAK:HIGH:SDEViation
value: List[float] = driver.multiEval.listPy.modulation.merror.peak.high.
↪standardDev.fetch()
```

Return magnitude error peak values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
mag_err_peak_high: float Comma-separated list of values, one per measured segment
Unit: %

6.2.9.4.7.18 Low

class LowCls

Low commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.merror.peak.low.clone()
```

Subgroups

6.2.9.4.7.19 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:PEAK:LOW:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:PEAK:LOW:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:PEAK:LOW:AVERage
value: List[float or bool] = driver.multiEval.listPy.modulation.merror.peak.low.
↪average.calculate()
```

Return magnitude error peak values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
mag_error_peak_low: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: %

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:PEAK:LOW:AVERage
value: List[float] = driver.multiEval.listPy.modulation.merror.peak.low.average.
↪fetch()
```

Return magnitude error peak values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
mag_error_peak_low: float Comma-separated list of values, one per measured segment Unit: %

6.2.9.4.7.20 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:PEAK:LOW:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:PEAK:LOW:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:PEAK:LOW:CURRent
value: List[float or bool] = driver.multiEval.listPy.modulation.merror.peak.low.
↪current.calculate()
```

Return magnitude error peak values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
mag_error_peak_low: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: %

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:PEAK:LOW:CURRent
value: List[float] = driver.multiEval.listPy.modulation.merror.peak.low.current.
↪fetch()
```

Return magnitude error peak values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
mag_error_peak_low: float Comma-separated list of values, one per measured segment Unit: %

6.2.9.4.7.21 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:PEAK:LOW:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:PEAK:LOW:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:PEAK:LOW:EXTReme
value: List[float or bool] = driver.multiEval.listPy.modulation.merror.peak.low.
↪extreme.calculate()
```

Return magnitude error peak values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
mag_error_peak_low: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: %

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:PEAK:LOW:EXTReme
value: List[float] = driver.multiEval.listPy.modulation.merror.peak.low.extreme.
↪fetch()
```

Return magnitude error peak values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
mag_error_peak_low: float Comma-separated list of values, one per measured segment Unit: %

6.2.9.4.7.22 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:PEAK:LOW:SDEViation
```

class StandardDevCls

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

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:PEAK:LOW:SDEViation
value: List[float] = driver.multiEval.listPy.modulation.merror.peak.low.
↪standardDev.fetch()
```

Return magnitude error peak values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
mag_error_peak_low: float Comma-separated list of values, one per measured segment Unit: %

6.2.9.4.7.23 Rms

class RmsCls

Rms commands group definition. 14 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.merror.rms.clone()
```

Subgroups

6.2.9.4.7.24 High

class HighCls

High commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.merror.rms.high.clone()
```

Subgroups

6.2.9.4.7.25 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:RMS:HIGH:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:RMS:HIGH:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:RMS:HIGH:AVERage
value: List[float or bool] = driver.multiEval.listPy.modulation.merror.rms.high.
↪average.calculate()
```

Return magnitude error RMS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
mag_error_rms_high: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: %

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:RMS:HIGH:AVERage
value: List[float] = driver.multiEval.listPy.modulation.merror.rms.high.average.
↪fetch()
```

Return magnitude error RMS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
mag_error_rms_high: float Comma-separated list of values, one per measured segment Unit: %

6.2.9.4.7.26 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:RMS:HIGH:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:RMS:HIGH:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:RMS:HIGH:CURRent
value: List[float or bool] = driver.multiEval.listPy.modulation.merror.rms.high.
↪current.calculate()
```

Return magnitude error RMS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
mag_error_rms_high: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: %

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:RMS:HIGH:CURRent
value: List[float] = driver.multiEval.listPy.modulation.merror.rms.high.current.
↪fetch()
```

Return magnitude error RMS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
mag_error_rms_high: float Comma-separated list of values, one per measured segment Unit: %

6.2.9.4.7.27 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:RMS:HIGH:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:RMS:HIGH:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:RMS:HIGH:EXTReme
value: List[float or bool] = driver.multiEval.listPy.modulation.merror.rms.high.
↪extreme.calculate()
```

Return magnitude error RMS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
mag_error_rms_high: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: %

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:RMS:HIGH:EXTReme
value: List[float] = driver.multiEval.listPy.modulation.merror.rms.high.extreme.
↪fetch()
```

Return magnitude error RMS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
mag_error_rms_high: float Comma-separated list of values, one per measured segment Unit: %

6.2.9.4.7.28 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:RMS:HIGH:SDEViation
```

class StandardDevCls

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

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:RMS:HIGH:SDEViation
value: List[float] = driver.multiEval.listPy.modulation.merror.rms.high.
↪standardDev.fetch()
```

Return magnitude error RMS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
mag_error_rms_high: float Comma-separated list of values, one per measured segment Unit: %

6.2.9.4.7.29 Low

class LowCls

Low commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.merror.rms.low.clone()
```

Subgroups

6.2.9.4.7.30 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:RMS:LOW:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:RMS:LOW:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:RMS:LOW:AVERage
value: List[float or bool] = driver.multiEval.listPy.modulation.merror.rms.low.
↪average.calculate()
```

Return magnitude error RMS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
mag_error_rms_low: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: %

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:RMS:LOW:AVERage
value: List[float] = driver.multiEval.listPy.modulation.merror.rms.low.average.
↪fetch()
```

Return magnitude error RMS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
mag_error_rms_low: float Comma-separated list of values, one per measured segment
Unit: %

6.2.9.4.7.31 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:RMS:LOW:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:RMS:LOW:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:RMS:LOW:CURRent
value: List[float or bool] = driver.multiEval.listPy.modulation.merror.rms.low.
↪current.calculate()
```

Return magnitude error RMS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
mag_error_rms_low: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: %

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:RMS:LOW:CURRent
value: List[float] = driver.multiEval.listPy.modulation.merror.rms.low.current.
↪fetch()
```

Return magnitude error RMS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
mag_error_rms_low: float Comma-separated list of values, one per measured segment
Unit: %

6.2.9.4.7.32 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:RMS:LOW:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:RMS:LOW:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:RMS:LOW:EXTReme
value: List[float or bool] = driver.multiEval.listPy.modulation.merror.rms.low.
↪extreme.calculate()
```

Return magnitude error RMS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
mag_error_rms_low: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: %

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:RMS:LOW:EXTReme
value: List[float] = driver.multiEval.listPy.modulation.merror.rms.low.extreme.
↪fetch()
```

Return magnitude error RMS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
mag_error_rms_low: float Comma-separated list of values, one per measured segment Unit: %

6.2.9.4.7.33 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:RMS:LOW:SDEViation
```

class StandardDevCls

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

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:RMS:LOW:SDEViation
value: List[float] = driver.multiEval.listPy.modulation.merror.rms.low.
↪standardDev.fetch()
```

Return magnitude error RMS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
mag_error_rms_low: float Comma-separated list of values, one per measured segment
Unit: %

6.2.9.4.8 Perror

class PerrorCls

Perror commands group definition. 42 total commands, 3 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.9.4.8.1 Dmrs

class DmrsCls

Dmrs commands group definition. 14 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.perror.dmrs.clone()
```

Subgroups

6.2.9.4.8.2 High

class HighCls

High commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.perror.dmrs.high.clone()
```

Subgroups

6.2.9.4.8.3 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:DMRS:HIGH:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:DMRS:HIGH:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:DMRS:HIGH:AVERage
value: List[float or bool] = driver.multiEval.listPy.modulation.perror.dmrs.
↪high.average.calculate()
```

Return phase error DMRS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ph_error_dmrs_high: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: deg

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:DMRS:HIGH:AVERage
value: List[float] = driver.multiEval.listPy.modulation.perror.dmrs.high.
↪average.fetch()
```

Return phase error DMRS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ph_error_dmrs_high: float Comma-separated list of values, one per measured segment Unit: deg

6.2.9.4.8.4 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:DMRS:HIGH:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:DMRS:HIGH:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:DMRS:HIGH:CURRent
value: List[float or bool] = driver.multiEval.listPy.modulation.perror.dmrs.
↪high.current.calculate()
```

Return phase error DMRS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ph_error_dmrs_high: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: deg

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:DMRS:HIGH:CURRent
value: List[float] = driver.multiEval.listPy.modulation.perror.dmrs.high.
↪current.fetch()
```

Return phase error DMRS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ph_error_dmrs_high: float Comma-separated list of values, one per measured segment Unit: deg

6.2.9.4.8.5 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:DMRS:HIGH:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:DMRS:HIGH:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:DMRS:HIGH:EXTReme
value: List[float or bool] = driver.multiEval.listPy.modulation.perror.dmrs.
↪high.extreme.calculate()
```

Return phase error DMRS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ph_error_dmrs_high: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: deg

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:DMRS:HIGH:EXTReme
value: List[float] = driver.multiEval.listPy.modulation.perror.dmrs.high.
↪extreme.fetch()
```

Return phase error DMRS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ph_error_dmrs_high: float Comma-separated list of values, one per measured segment Unit: deg

6.2.9.4.8.6 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:DMRS:HIGH:SDEViation
```

class StandardDevCls

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

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:DMRS:HIGH:SDEViation
value: List[float] = driver.multiEval.listPy.modulation.perror.dmrs.high.
↪standardDev.fetch()
```

Return phase error DMRS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ph_error_dmrs_high: float Comma-separated list of values, one per measured segment
Unit: deg

6.2.9.4.8.7 Low

class LowCls

Low commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.perror.dmrs.low.clone()
```

Subgroups

6.2.9.4.8.8 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:DMRS:LOW:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:DMRS:LOW:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:DMRS:LOW:AVERage
value: List[float or bool] = driver.multiEval.listPy.modulation.perror.dmrs.low.
↪average.calculate()
```

Return phase error DMRS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ph_error_dmrs_low: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: deg

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:DMRS:LOW:AVERage
value: List[float] = driver.multiEval.listPy.modulation.perror.dmrs.low.average.
↪fetch()
```

Return phase error DMRS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ph_error_dmrs_low: float Comma-separated list of values, one per measured segment
Unit: deg

6.2.9.4.8.9 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:DMRS:LOW:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:DMRS:LOW:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:DMRS:LOW:CURRent
value: List[float or bool] = driver.multiEval.listPy.modulation.perror.dmrs.low.
↪current.calculate()
```

Return phase error DMRS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ph_error_dmrs_low: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: deg

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:DMRS:LOW:CURRent
value: List[float] = driver.multiEval.listPy.modulation.perror.dmrs.low.current.
↪fetch()
```

Return phase error DMRS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ph_error_dmrs_low: float Comma-separated list of values, one per measured segment
Unit: deg

6.2.9.4.8.10 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:DMRS:LOW:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:DMRS:LOW:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:DMRS:LOW:EXTReme
value: List[float or bool] = driver.multiEval.listPy.modulation.perror.dmrs.low.
↪extreme.calculate()
```

Return phase error DMRS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ph_error_dmrs_low: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: deg

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:DMRS:LOW:EXTReme
value: List[float] = driver.multiEval.listPy.modulation.perror.dmrs.low.extreme.
↪fetch()
```

Return phase error DMRS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ph_error_dmrs_low: float Comma-separated list of values, one per measured segment Unit: deg

6.2.9.4.8.11 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:DMRS:LOW:SDEViation
```

class StandardDevCls

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

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:DMRS:LOW:SDEViation
value: List[float] = driver.multiEval.listPy.modulation.perror.dmrs.low.
↪standardDev.fetch()
```

Return phase error DMRS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ph_error_dmrs_low: float Comma-separated list of values, one per measured segment
Unit: deg

6.2.9.4.8.12 Peak

class PeakCls

Peak commands group definition. 14 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.perror.peak.clone()
```

Subgroups

6.2.9.4.8.13 High

class HighCls

High commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.perror.peak.high.clone()
```

Subgroups

6.2.9.4.8.14 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:PEAK:HIGH:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:PEAK:HIGH:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:PEAK:HIGH:AVERage
value: List[float or bool] = driver.multiEval.listPy.modulation.perror.peak.
↪high.average.calculate()
```

Return phase error peak values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ph_error_peak_high: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: deg

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:PEAK:HIGH:AVERage
value: List[float] = driver.multiEval.listPy.modulation.perror.peak.high.
↪average.fetch()
```

Return phase error peak values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ph_error_peak_high: float Comma-separated list of values, one per measured segment Unit: deg

6.2.9.4.8.15 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:PEAK:HIGH:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:PEAK:HIGH:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:PEAK:HIGH:CURRent
value: List[float or bool] = driver.multiEval.listPy.modulation.perror.peak.
↪high.current.calculate()
```

Return phase error peak values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ph_error_peak_high: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: deg

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:PEAK:HIGH:CURRent
value: List[float] = driver.multiEval.listPy.modulation.perror.peak.high.
↪current.fetch()
```

Return phase error peak values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ph_error_peak_high: float Comma-separated list of values, one per measured segment Unit: deg

6.2.9.4.8.16 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:PEAK:HIGH:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:PEAK:HIGH:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:PEAK:HIGH:EXTReme
value: List[float or bool] = driver.multiEval.listPy.modulation.perror.peak.
↪high.extreme.calculate()
```

Return phase error peak values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ph_error_peak_high: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: deg

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:PEAK:HIGH:EXTReme
value: List[float] = driver.multiEval.listPy.modulation.perror.peak.high.
↪extreme.fetch()
```

Return phase error peak values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ph_error_peak_high: float Comma-separated list of values, one per measured segment
Unit: deg

6.2.9.4.8.17 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:PEAK:HIGH:SDEViation
```

class StandardDevCls

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

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:PEAK:HIGH:SDEViation
value: List[float] = driver.multiEval.listPy.modulation.perror.peak.high.
↪standardDev.fetch()
```

Return phase error peak values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ph_error_peak_high: float Comma-separated list of values, one per measured segment
Unit: deg

6.2.9.4.8.18 Low

class LowCls

Low commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.perror.peak.low.clone()
```

Subgroups

6.2.9.4.8.19 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:PEAK:LOW:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:PEAK:LOW:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:PEAK:LOW:AVERage
value: List[float or bool] = driver.multiEval.listPy.modulation.perror.peak.low.
↪average.calculate()
```

Return phase error peak values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ph_error_peak_low: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: deg

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:PEAK:LOW:AVERage
value: List[float] = driver.multiEval.listPy.modulation.perror.peak.low.average.
↪fetch()
```

Return phase error peak values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ph_error_peak_low: float Comma-separated list of values, one per measured segment Unit: deg

6.2.9.4.8.20 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:PEAK:LOW:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:PEAK:LOW:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:PEAK:LOW:CURRent
value: List[float or bool] = driver.multiEval.listPy.modulation.perror.peak.low.
↪current.calculate()
```

Return phase error peak values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ph_error_peak_low: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: deg

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:PEAK:LOW:CURRent
value: List[float] = driver.multiEval.listPy.modulation.perror.peak.low.current.
↪fetch()
```

Return phase error peak values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ph_error_peak_low: float Comma-separated list of values, one per measured segment Unit: deg

6.2.9.4.8.21 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:PEAK:LOW:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:PEAK:LOW:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:PEAK:LOW:EXTReme
value: List[float or bool] = driver.multiEval.listPy.modulation.perror.peak.low.
↪extreme.calculate()
```

Return phase error peak values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ph_error_peak_low: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: deg

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:PEAK:LOW:EXTReme
value: List[float] = driver.multiEval.listPy.modulation.perror.peak.low.extreme.
↪fetch()
```

Return phase error peak values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ph_error_peak_low: float Comma-separated list of values, one per measured segment Unit: deg

6.2.9.4.8.22 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:PEAK:LOW:SDEViation
```

class StandardDevCls

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

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:PEAK:LOW:SDEViation
value: List[float] = driver.multiEval.listPy.modulation.perror.peak.low.
↪standardDev.fetch()
```

Return phase error peak values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ph_error_peak_low: float Comma-separated list of values, one per measured segment
Unit: deg

6.2.9.4.8.23 Rms

class RmsCls

Rms commands group definition. 14 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.perror.rms.clone()
```

Subgroups

6.2.9.4.8.24 High

class HighCls

High commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.perror.rms.high.clone()
```

Subgroups

6.2.9.4.8.25 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:RMS:HIGH:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:RMS:HIGH:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:RMS:HIGH:AVERage
value: List[float or bool] = driver.multiEval.listPy.modulation.perror.rms.high.
↪average.calculate()
```

Return phase error RMS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ph_error_rms_high: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: deg

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:RMS:HIGH:AVERage
value: List[float] = driver.multiEval.listPy.modulation.perror.rms.high.average.
↪fetch()
```

Return phase error RMS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ph_error_rms_high: float Comma-separated list of values, one per measured segment Unit: deg

6.2.9.4.8.26 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:RMS:HIGH:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:RMS:HIGH:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:RMS:HIGH:CURRent
value: List[float or bool] = driver.multiEval.listPy.modulation.perror.rms.high.
↪current.calculate()
```

Return phase error RMS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ph_error_rms_high: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: deg

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:RMS:HIGH:CURRent
value: List[float] = driver.multiEval.listPy.modulation.perror.rms.high.current.
↪fetch()
```

Return phase error RMS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ph_error_rms_high: float Comma-separated list of values, one per measured segment
Unit: deg

6.2.9.4.8.27 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:RMS:HIGH:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:RMS:HIGH:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:RMS:HIGH:EXTReme
value: List[float or bool] = driver.multiEval.listPy.modulation.perror.rms.high.
↪extreme.calculate()
```

Return phase error RMS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ph_error_rms_high: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: deg

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:RMS:HIGH:EXTReme
value: List[float] = driver.multiEval.listPy.modulation.perror.rms.high.extreme.
↪fetch()
```

Return phase error RMS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ph_error_rms_high: float Comma-separated list of values, one per measured segment
Unit: deg

6.2.9.4.8.28 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:RMS:HIGH:SDEViation
```

class StandardDevCls

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

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:RMS:HIGH:SDEViation
value: List[float] = driver.multiEval.listPy.modulation.perror.rms.high.
↪standardDev.fetch()
```

Return phase error RMS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ph_error_rms_high: float Comma-separated list of values, one per measured segment
Unit: deg

6.2.9.4.8.29 Low

class LowCls

Low commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.perror.rms.low.clone()
```

Subgroups

6.2.9.4.8.30 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:RMS:LOW:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:RMS:LOW:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:RMS:LOW:AVERage
value: List[float or bool] = driver.multiEval.listPy.modulation.perror.rms.low.
↪average.calculate()
```

Return phase error RMS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ph_error_rms_low: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: deg

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:RMS:LOW:AVERage
value: List[float] = driver.multiEval.listPy.modulation.perror.rms.low.average.
↪fetch()
```

Return phase error RMS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ph_error_rms_low: float Comma-separated list of values, one per measured segment Unit: deg

6.2.9.4.8.31 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:RMS:LOW:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:RMS:LOW:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:RMS:LOW:CURRent
value: List[float or bool] = driver.multiEval.listPy.modulation.perror.rms.low.
↪current.calculate()
```

Return phase error RMS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ph_error_rms_low: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: deg

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:RMS:LOW:CURRent
value: List[float] = driver.multiEval.listPy.modulation.perror.rms.low.current.
↪fetch()
```

Return phase error RMS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ph_error_rms_low: float Comma-separated list of values, one per measured segment Unit: deg

6.2.9.4.8.32 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:RMS:LOW:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:RMS:LOW:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:RMS:LOW:EXTReme
value: List[float or bool] = driver.multiEval.listPy.modulation.perror.rms.low.
↪extreme.calculate()
```

Return phase error RMS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ph_error_rms_low: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: deg

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:RMS:LOW:EXTReme
value: List[float] = driver.multiEval.listPy.modulation.perror.rms.low.extreme.
↪fetch()
```

Return phase error RMS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ph_error_rms_low: float Comma-separated list of values, one per measured segment
Unit: deg

6.2.9.4.8.33 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:RMS:LOW:SDEViation
```

class StandardDevCls

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

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:RMS:LOW:SDEViation
value: List[float] = driver.multiEval.listPy.modulation.perror.rms.low.
↪standardDev.fetch()
```

Return phase error RMS values for low and high EVM window position, for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ph_error_rms_low: float Comma-separated list of values, one per measured segment
Unit: deg

6.2.9.4.9 Ppower

class PpowerCls

Ppower commands group definition. 9 total commands, 5 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.9.4.9.1 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PPOWer:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PPOWer:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PPOWer:AVERage
value: List[float or bool] = driver.multiEval.listPy.modulation.ppower.average.
↪calculate()
```

Return user equipment peak power values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
peak_power: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: dBm

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PPOWer:AVERage
value: List[float] = driver.multiEval.listPy.modulation.ppower.average.fetch()
```

Return user equipment peak power values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
peak_power: float Comma-separated list of values, one per measured segment Unit: dBm

6.2.9.4.9.2 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PPOWer:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PPOWer:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PPOWer:CURRent
value: List[float or bool] = driver.multiEval.listPy.modulation.ppower.current.
↪calculate()
```

Return user equipment peak power values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
peak_power: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: dBm

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PPOWer:CURRent
value: List[float] = driver.multiEval.listPy.modulation.ppower.current.fetch()
```

Return user equipment peak power values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
peak_power: float Comma-separated list of values, one per measured segment Unit: dBm

6.2.9.4.9.3 Maximum

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PPOWer:MAXimum
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PPOWer:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PPOWer:MAXimum
value: List[float or bool] = driver.multiEval.listPy.modulation.ppower.maximum.
↪calculate()
```

Return user equipment peak power values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
peak_power: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: dBm

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PPOWer:MAXimum
value: List[float] = driver.multiEval.listPy.modulation.ppower.maximum.fetch()
```

Return user equipment peak power values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
peak_power: float Comma-separated list of values, one per measured segment Unit: dBm

6.2.9.4.9.4 Minimum

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PPOWer:MINimum
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PPOWer:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PPOWer:MINimum
value: List[float or bool] = driver.multiEval.listPy.modulation.ppower.minimum.
↪calculate()
```

Return user equipment peak power values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
peak_power: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: dBm

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PPOWer:MINimum
value: List[float] = driver.multiEval.listPy.modulation.ppower.minimum.fetch()
```

Return user equipment peak power values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
peak_power: float Comma-separated list of values, one per measured segment Unit: dBm

6.2.9.4.9.5 StandardDev

SCPI Command :

```
FETCh:LTE: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:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PPOWer:SDEViation
value: List[float] = driver.multiEval.listPy.modulation.ppower.standardDev.
↪fetch()
```

Return user equipment peak power values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
peak_power: float Comma-separated list of values, one per measured segment Unit: dBm

6.2.9.4.10 Psd

class PsdCls

Psd commands group definition. 9 total commands, 5 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.9.4.10.1 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PSD:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PSD:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PSD:AVERage
value: List[float or bool] = driver.multiEval.listPy.modulation.psd.average.
↪calculate()
```

Return RB power values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
psd: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: dBm

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PSD:AVERage
value: List[float] = driver.multiEval.listPy.modulation.psd.average.fetch()
```

Return RB power values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
psd: float Comma-separated list of values, one per measured segment Unit: dBm

6.2.9.4.10.2 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PSD:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PSD:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PSD:CURRent
value: List[float or bool] = driver.multiEval.listPy.modulation.psd.current.
↪calculate()
```

Return RB power values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
psd: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: dBm

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PSD:CURRent
value: List[float] = driver.multiEval.listPy.modulation.psd.current.fetch()
```

Return RB power values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
psd: float Comma-separated list of values, one per measured segment Unit: dBm

6.2.9.4.10.3 Maximum

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PSD:MAXimum
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PSD:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PSD:MAXimum
value: List[float or bool] = driver.multiEval.listPy.modulation.psd.maximum.
↪calculate()
```

Return RB power values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
psd: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: dBm

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PSD:MAXimum
value: List[float] = driver.multiEval.listPy.modulation.psd.maximum.fetch()
```

Return RB power values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
psd: float Comma-separated list of values, one per measured segment Unit: dBm

6.2.9.4.10.4 Minimum

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PSD:MINimum
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PSD:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PSD:MINimum
value: List[float or bool] = driver.multiEval.listPy.modulation.psd.minimum.
↪calculate()
```

Return RB power values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
psd: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: dBm

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PSD:MINimum
value: List[float] = driver.multiEval.listPy.modulation.psd.minimum.fetch()
```

Return RB power values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
psd: float Comma-separated list of values, one per measured segment Unit: dBm

6.2.9.4.10.5 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PSD:SDEViation
```

class StandardDevCls

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

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PSD:SDEViation
value: List[float] = driver.multiEval.listPy.modulation.psd.standardDev.fetch()
```

Return RB power values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
psd: float Comma-separated list of values, one per measured segment Unit: dBm

6.2.9.4.11 SchType

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:SCHType
```

class SchTypeCls

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

fetch() → List[SidelinkChannelType]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:SCHType
value: List[enums.SidelinkChannelType] = driver.multiEval.listPy.modulation.
↪schType.fetch()
```

Returns the sidelink channel type evaluated for modulation results, for all measured list mode segments.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
channel_type: PSSCh | PSCCh Comma-separated list of values, one per measured segment

6.2.9.4.12 Terror

class TerrorCls

Terror commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.9.4.12.1 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:TERRor:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:TERRor:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:TERRor:AVERage
value: List[float or bool] = driver.multiEval.listPy.modulation.terror.average.
↪calculate()
```

Return transmit time error values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
timing_error: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: Ts (basic LTE time unit)

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:TERRor:AVERage
value: List[float] = driver.multiEval.listPy.modulation.terror.average.fetch()
```

Return transmit time error values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
timing_error: float Comma-separated list of values, one per measured segment Unit: Ts (basic LTE time unit)

6.2.9.4.12.2 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:TERRor:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:TERRor:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:TERRor:CURRent
value: List[float or bool] = driver.multiEval.listPy.modulation.terror.current.
↪calculate()
```

Return transmit time error values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
timing_error: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: Ts (basic LTE time unit)

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:TERRor:CURRent
value: List[float] = driver.multiEval.listPy.modulation.terror.current.fetch()
```

Return transmit time error values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
timing_error: float Comma-separated list of values, one per measured segment Unit: Ts (basic LTE time unit)

6.2.9.4.12.3 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:TERRor:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:TERRor:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:TERRor:EXTReme
value: List[float or bool] = driver.multiEval.listPy.modulation.terror.extreme.
↪calculate()
```

Return transmit time error values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
timing_error: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: Ts (basic LTE time unit)

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:TERRor:EXTReme
value: List[float] = driver.multiEval.listPy.modulation.terror.extreme.fetch()
```

Return transmit time error values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
timing_error: float Comma-separated list of values, one per measured segment Unit: Ts (basic LTE time unit)

6.2.9.4.12.4 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:TERRor:SDEViation
```

class StandardDevCls

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

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:TERRor:SDEViation
value: List[float] = driver.multiEval.listPy.modulation.terror.standardDev.
↪fetch()
```

Return transmit time error values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
timing_error: float Comma-separated list of values, one per measured segment Unit: Ts (basic LTE time unit)

6.2.9.4.13 Tpower

class TpowerCls

Tpower commands group definition. 9 total commands, 5 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.9.4.13.1 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:TPOWer:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:TPOWer:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:TPOWer:AVERage
value: List[float or bool] = driver.multiEval.listPy.modulation.tpower.average.
↪calculate()
```

Return user equipment power values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
tx_power: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: dBm

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:TPOWer:AVERage
value: List[float] = driver.multiEval.listPy.modulation.tpower.average.fetch()
```

Return user equipment power values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
tx_power: float Comma-separated list of values, one per measured segment Unit: dBm

6.2.9.4.13.2 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:TPOWer:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:TPOWer:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:TPOWer:CURRent
value: List[float or bool] = driver.multiEval.listPy.modulation.tpower.current.
↪calculate()
```

Return user equipment power values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
tx_power: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: dBm

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:TPOWer:CURRent
value: List[float] = driver.multiEval.listPy.modulation.tpower.current.fetch()
```

Return user equipment power values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
tx_power: float Comma-separated list of values, one per measured segment Unit: dBm

6.2.9.4.13.3 Maximum

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:TPOWer:MAXimum
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:TPOWer:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:TPOWer:MAXimum
value: List[float or bool] = driver.multiEval.listPy.modulation.tpower.maximum.
↪calculate()
```

Return user equipment power values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
tx_power: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: dBm

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:TPOWer:MAXimum
value: List[float] = driver.multiEval.listPy.modulation.tpower.maximum.fetch()
```

Return user equipment power values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
tx_power: float Comma-separated list of values, one per measured segment Unit: dBm

6.2.9.4.13.4 Minimum

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:TPOWer:MINimum
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:TPOWer:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:TPOWer:MINimum
value: List[float or bool] = driver.multiEval.listPy.modulation.tpower.minimum.
↪calculate()
```

Return user equipment power values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
tx_power: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: dBm

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:TPOWer:MINimum
value: List[float] = driver.multiEval.listPy.modulation.tpower.minimum.fetch()
```

Return user equipment power values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
tx_power: float Comma-separated list of values, one per measured segment Unit: dBm

6.2.9.4.13.5 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:TPOWer:SDEViation
```

class StandardDevCls

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

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:TPOWer:SDEViation
value: List[float] = driver.multiEval.listPy.modulation.tpower.standardDev.
↪fetch()
```

Return user equipment power values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
tx_power: float Comma-separated list of values, one per measured segment Unit: dBm

6.2.9.5 Pmonitor

class PmonitorCls

Pmonitor 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.multiEval.listPy.pmonitor.clone()
```

Subgroups

6.2.9.5.1 Peak

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:PMONitor:PEAK
```

class PeakCls

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

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:PMONitor:PEAK
value: List[float] = driver.multiEval.listPy.pmonitor.peak.fetch()
```

Return the power monitor results for all measured segments in list mode. The commands return one power result per subframe for the measured carrier. The power values are RMS averaged over the subframe or represent the peak value within the subframe.

INTRO_CMD_HELP: Commands for querying the result list structure:

- method RsCmwLteMeas.MultiEval.ListPy.Segment.Pmonitor.Array.Start.fetch
- method RsCmwLteMeas.MultiEval.ListPy.Segment.Pmonitor.Array.Length.fetch

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
step_peak_power: float Comma-separated list of power values, one value per subframe, from first subframe of first measured segment to last subframe of last measured segment For an inactive segment only one INV is returned, independent of the number of configured subframes. Unit: dBm

6.2.9.5.2 Rms

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:PMONitor:RMS
```

class RmsCls

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

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:PMONitor:RMS
value: List[float] = driver.multiEval.listPy.pmonitor.rms.fetch()
```

Return the power monitor results for all measured segments in list mode. The commands return one power result per subframe for the measured carrier. The power values are RMS averaged over the subframe or represent the peak value within the subframe.

INTRO_CMD_HELP: Commands for querying the result list structure:

- method RsCmwLteMeas.MultiEval.ListPy.Segment.Pmonitor.Array.Start.fetch
- method RsCmwLteMeas.MultiEval.ListPy.Segment.Pmonitor.Array.Length.fetch

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
step_rms_power: float Comma-separated list of power values, one value per subframe, from first subframe of first measured segment to last subframe of last measured segment For an inactive segment only one INV is returned, independent of the number of configured subframes. Unit: dBm

6.2.9.6 Power

class PowerCls

Power 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.multiEval.listPy.power.clone()
```

Subgroups

6.2.9.6.1 TxPower

class TxPowerCls

TxPower commands group definition. 9 total commands, 5 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.9.6.1.1 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:POWer:TXPower:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:POWer:TXPower:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:POWer:TXPower:AVERage
value: List[float or bool] = driver.multiEval.listPy.power.txPower.average.
↪calculate()
```

No command help available

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
tx_power: (float or boolean items) No help available

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:POWer:TXPower:AVERage
value: List[float] = driver.multiEval.listPy.power.txPower.average.fetch()
```

Return the total TX power of all component carriers, for all measured list mode segments.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
tx_power: float Comma-separated list of values, one per measured segment Unit: dBm

6.2.9.6.1.2 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:POWer:TXPower:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:POWer:TXPower:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:POWer:TXPower:CURRent
value: List[float or bool] = driver.multiEval.listPy.power.txPower.current.
↪calculate()
```

No command help available

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
tx_power: (float or boolean items) No help available

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:POWer:TXPower:CURRent
value: List[float] = driver.multiEval.listPy.power.txPower.current.fetch()
```

Return the total TX power of all component carriers, for all measured list mode segments.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
tx_power: float Comma-separated list of values, one per measured segment Unit: dBm

6.2.9.6.1.3 Maximum

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:POWer:TXPower:MAXimum
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:POWer:TXPower:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:POWer:TXPower:MAXimum
value: List[float or bool] = driver.multiEval.listPy.power.txPower.maximum.
↪calculate()
```

No command help available

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
tx_power: (float or boolean items) No help available

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:POWer:TXPower:MAXimum
value: List[float] = driver.multiEval.listPy.power.txPower.maximum.fetch()
```

Return the total TX power of all component carriers, for all measured list mode segments.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
tx_power: float Comma-separated list of values, one per measured segment Unit: dBm

6.2.9.6.1.4 Minimum

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:POWer:TXPower:MINimum
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:POWer:TXPower:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:POWer:TXPower:MINimum
value: List[float or bool] = driver.multiEval.listPy.power.txPower.minimum.
↪calculate()
```

No command help available

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
tx_power: (float or boolean items) No help available

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:POWer:TXPower:MINimum
value: List[float] = driver.multiEval.listPy.power.txPower.minimum.fetch()
```

Return the total TX power of all component carriers, for all measured list mode segments.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
tx_power: float Comma-separated list of values, one per measured segment Unit: dBm

6.2.9.6.1.5 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:POWer:TXPower:SDEViation
```

class StandardDevCls

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

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:POWer:TXPower:SDEViation
value: List[float] = driver.multiEval.listPy.power.txPower.standardDev.fetch()
```

Return the total TX power of all component carriers, for all measured list mode segments.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
tx_power: float Comma-separated list of values, one per measured segment Unit: dBm

6.2.9.7 Segment

RepCap Settings

```
# Range: Nr1 .. Nr128
rc = driver.multiEval.listPy.segment.repcap_segment_get()
driver.multiEval.listPy.segment.repcap_segment_set(repcap.Segment.Nr1)
```

class SegmentCls

Segment commands group definition. 81 total commands, 7 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.multiEval.listPy.segment.clone()
```

Subgroups

6.2.9.7.1 Aclr

class AclrCls

Aclr commands group definition. 6 total commands, 4 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.9.7.1.1 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:ACLR:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:ACLR:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in slots
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Utra_2_Neg: enums.ResultStatus2: float ACLR for the second UTRA channel with lower frequency Unit: dB
- Utra_1_Neg: enums.ResultStatus2: float ACLR for the first UTRA channel with lower frequency Unit: dB
- Eutra_Negativ: enums.ResultStatus2: float ACLR for the first E-UTRA channel below the carrier frequency Unit: dB
- Eutra: enums.ResultStatus2: float Power in the allocated E-UTRA channel Unit: dBm
- Eutra_Positiv: enums.ResultStatus2: float ACLR for the first E-UTRA channel above the carrier frequency Unit: dB
- Utra_1_Pos: enums.ResultStatus2: float ACLR for the first UTRA channel with higher frequency Unit: dB
- Utra_2_Pos: enums.ResultStatus2: float ACLR for the second UTRA channel with higher frequency Unit: dB

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in slots
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Utra_2_Neg: float: float ACLR for the second UTRA channel with lower frequency Unit: dB
- Utra_1_Neg: float: float ACLR for the first UTRA channel with lower frequency Unit: dB
- Eutra_Negativ: float: float ACLR for the first E-UTRA channel below the carrier frequency Unit: dB
- Eutra: float: float Power in the allocated E-UTRA channel Unit: dBm
- Eutra_Positiv: float: float ACLR for the first E-UTRA channel above the carrier frequency Unit: dB
- Utra_1_Pos: float: float ACLR for the first UTRA channel with higher frequency Unit: dB
- Utra_2_Pos: float: float ACLR for the second UTRA channel with higher frequency Unit: dB

calculate(*segment=Segment.Default*) → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:ACLR:AVERage
value: CalculateStruct = driver.multiEval.listPy.segment.aclr.average.
↪calculate(segment = repcap.Segment.Default)
```

Return ACLR single value results for segment <no> in list mode. The values described below are returned by FETCh commands. The first four values (reliability to out-of-tolerance result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:ACLR:AVERage
value: FetchStruct = driver.multiEval.listPy.segment.aclr.average.fetch(segment␣
↪= repcap.Segment.Default)
```

Return ACLR single value results for segment <no> in list mode. The values described below are returned by FETCh commands. The first four values (reliability to out-of-tolerance result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

6.2.9.7.1.2 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:ACLR:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:ACLR:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in slots
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Utra_2_Neg: enums.ResultStatus2: float ACLR for the second UTRA channel with lower frequency Unit: dB
- Utra_1_Neg: enums.ResultStatus2: float ACLR for the first UTRA channel with lower frequency Unit: dB

- Eutra_Negativ: enums.ResultStatus2: float ACLR for the first E-UTRA channel below the carrier frequency Unit: dB
- Eutra: enums.ResultStatus2: float Power in the allocated E-UTRA channel Unit: dBm
- Eutra_Positiv: enums.ResultStatus2: float ACLR for the first E-UTRA channel above the carrier frequency Unit: dB
- Utra_1_Pos: enums.ResultStatus2: float ACLR for the first UTRA channel with higher frequency Unit: dB
- Utra_2_Pos: enums.ResultStatus2: float ACLR for the second UTRA channel with higher frequency Unit: dB

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in slots
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Utra_2_Neg: float: float ACLR for the second UTRA channel with lower frequency Unit: dB
- Utra_1_Neg: float: float ACLR for the first UTRA channel with lower frequency Unit: dB
- Eutra_Negativ: float: float ACLR for the first E-UTRA channel below the carrier frequency Unit: dB
- Eutra: float: float Power in the allocated E-UTRA channel Unit: dBm
- Eutra_Positiv: float: float ACLR for the first E-UTRA channel above the carrier frequency Unit: dB
- Utra_1_Pos: float: float ACLR for the first UTRA channel with higher frequency Unit: dB
- Utra_2_Pos: float: float ACLR for the second UTRA channel with higher frequency Unit: dB

calculate(*segment=Segment.Default*) → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:ACLR:CURRent
value: CalculateStruct = driver.multiEval.listPy.segment.aclr.current.
↪calculate(segment = repcap.Segment.Default)
```

Return ACLR single value results for segment <no> in list mode. The values described below are returned by FETCh commands. The first four values (reliability to out-of-tolerance result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:ACLR:CURRent
```

(continues on next page)

(continued from previous page)

```
value: FetchStruct = driver.multiEval.listPy.segment.aclr.current.fetch(segment␣
↪= repcap.Segment.Default)
```

Return ACLR single value results for segment <no> in list mode. The values described below are returned by FETCh commands. The first four values (reliability to out-of-tolerance result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

6.2.9.7.1.3 Dallocation

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:ACLR:DALLocation
```

class DallocationCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Nr_Res_Blocks: int: decimal Number of allocated resource blocks
- Offset_Res_Blocks: int: decimal Offset of the first allocated resource block from the edge of the allocated UL transmission bandwidth

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:ACLR:DALLocation
value: FetchStruct = driver.multiEval.listPy.segment.aclr.dallocation.
↪fetch(segment = repcap.Segment.Default)
```

Return the detected allocation for segment <no> in list mode. The result is determined from the last measured slot of the statistical length. The individual measurements provide the same result when measuring the same slot. However different statistical lengths can be defined for the measurements so that the measured slots and returned results can differ.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

6.2.9.7.1.4 DchType

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:ACLR:DCHType
```

class DchTypeCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Channel_Type: enums.UplinkChannelType: PUSCh | PUCCh

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:ACLR:DCHType
value: FetchStruct = driver.multiEval.listPy.segment.aclr.dchType.fetch(segment␣
↪= repcap.Segment.Default)
```

Return the uplink channel type for segment <no> in list mode. The result is determined from the last measured slot of the statistical length. The individual measurements provide the same result when measuring the same slot. However different statistical lengths can be defined for the measurements so that the measured slots and returned results can differ.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

6.2.9.7.2 EsFlatness

class EsFlatnessCls

EsFlatness 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.multiEval.listPy.segment.esFlatness.clone()
```

Subgroups

6.2.9.7.2.1 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:ESFLatness:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:ESFLatness:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in slots
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Ripple_1: float or bool: Limit check result for max (range 1) - min (range 1) .
- Ripple_2: float or bool: Limit check result for max (range 2) - min (range 2) .
- Max_R_1_Min_R_2: float or bool: Limit check result for max (range 1) - min (range 2) . Unit: dB
- Max_R_2_Min_R_1: float or bool: Limit check result for max (range 2) - min (range 1) . Unit: dB

class FetchStruct

Response structure. Fields:

- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in slots
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Ripple_1: float: float Max (range 1) - min (range 1) Unit: dB
- Ripple_2: float: float Max (range 2) - min (range 2) Unit: dB
- Max_R_1_Min_R_2: float: float Max (range 1) - min (range 2) Unit: dB
- Max_R_2_Min_R_1: float: float Max (range 2) - min (range 1) Unit: dB
- Min_R_1: float: float Min (range 1) Unit: dB
- Max_R_1: float: float Max (range 1) Unit: dB
- Min_R_2: float: float Min (range 2) Unit: dB
- Max_R_2: float: float Max (range 2) Unit: dB

calculate(*segment=Segment.Default*) → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:ESFLatness:AVERage
value: CalculateStruct = driver.multiEval.listPy.segment.esFlatness.average.
↪calculate(segment = repcap.Segment.Default)
```

Return equalizer spectrum flatness single value results for segment <no> in list mode.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:ESFLatness:AVERage
value: FetchStruct = driver.multiEval.listPy.segment.esFlatness.average.
↪fetch(segment = repcap.Segment.Default)
```

Return equalizer spectrum flatness single value results for segment <no> in list mode.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

6.2.9.7.2.2 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:ESFLatness:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:ESFLatness:CURRent
```

class CurrentCls

Current commands group definition. 3 total commands, 1 Subgroups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in slots
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Ripple_1: float or bool: Limit check result for max (range 1) - min (range 1) .
- Ripple_2: float or bool: Limit check result for max (range 2) - min (range 2) .
- Max_R_1_Min_R_2: float or bool: Limit check result for max (range 1) - min (range 2) . Unit: dB
- Max_R_2_Min_R_1: float or bool: Limit check result for max (range 2) - min (range 1) . Unit: dB

class FetchStruct

Response structure. Fields:

- Seg_Reliability: int: decimal Reliability indicator for the segment

- Statist_Expired: int: decimal Reached statistical length in slots
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Ripple_1: float: float Max (range 1) - min (range 1) Unit: dB
- Ripple_2: float: float Max (range 2) - min (range 2) Unit: dB
- Max_R_1_Min_R_2: float: float Max (range 1) - min (range 2) Unit: dB
- Max_R_2_Min_R_1: float: float Max (range 2) - min (range 1) Unit: dB
- Min_R_1: float: float Min (range 1) Unit: dB
- Max_R_1: float: float Max (range 1) Unit: dB
- Min_R_2: float: float Min (range 2) Unit: dB
- Max_R_2: float: float Max (range 2) Unit: dB

calculate(*segment=Segment.Default*) → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:ESFLatness:CURRent
value: CalculateStruct = driver.multiEval.listPy.segment.esFlatness.current.
↪calculate(segment = repcap.Segment.Default)
```

Return equalizer spectrum flatness single value results for segment <no> in list mode.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:ESFLatness:CURRent
value: FetchStruct = driver.multiEval.listPy.segment.esFlatness.current.
↪fetch(segment = repcap.Segment.Default)
```

Return equalizer spectrum flatness single value results for segment <no> in list mode.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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.multiEval.listPy.segment.esFlatness.current.clone()
```

Subgroups

6.2.9.7.2.3 ScIndex

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:ESFLatness:CURRent:SCINdex
```

class ScIndexCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in slots
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Maximum_1: int: decimal SC index of max (range 1)
- Minimum_1: int: decimal SC index of min (range 1)
- Maximum_2: int: decimal SC index of max (range 2)
- Minimum_2: int: decimal SC index of min (range 2)

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:ESFLatness:CURRent:SCINdex
value: FetchStruct = driver.multiEval.listPy.segment.esFlatness.current.scIndex.
↪fetch(segment = repcap.Segment.Default)
```

Return subcarrier indices of the equalizer spectrum flatness measurement for segment <no> in list mode. At these SC indices, the current minimum and maximum power of the equalizer coefficients have been detected within range 1 and range 2.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

6.2.9.7.2.4 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:ESFLatness:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:ESFLatness:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in slots
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Ripple_1: float or bool: Limit check result for max (range 1) - min (range 1) .
- Ripple_2: float or bool: Limit check result for max (range 2) - min (range 2) .
- Max_R_1_Min_R_2: float or bool: Limit check result for max (range 1) - min (range 2) . Unit: dB
- Max_R_2_Min_R_1: float or bool: Limit check result for max (range 2) - min (range 1) . Unit: dB

class FetchStruct

Response structure. Fields:

- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in slots
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Ripple_1: float: float Max (range 1) - min (range 1) Unit: dB
- Ripple_2: float: float Max (range 2) - min (range 2) Unit: dB
- Max_R_1_Min_R_2: float: float Max (range 1) - min (range 2) Unit: dB
- Max_R_2_Min_R_1: float: float Max (range 2) - min (range 1) Unit: dB
- Min_R_1: float: float Min (range 1) Unit: dB
- Max_R_1: float: float Max (range 1) Unit: dB
- Min_R_2: float: float Min (range 2) Unit: dB
- Max_R_2: float: float Max (range 2) Unit: dB

calculate(*segment=Segment.Default*) → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:ESFLatness:EXTReme
value: CalculateStruct = driver.multiEval.listPy.segment.esFlatness.extreme.
↪calculate(segment = repcap.Segment.Default)
```

Return equalizer spectrum flatness single value results for segment <no> in list mode.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:ESFLatness:EXTReme
value: FetchStruct = driver.multiEval.listPy.segment.esFlatness.extreme.
↪fetch(segment = repcap.Segment.Default)
```

Return equalizer spectrum flatness single value results for segment <no> in list mode.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

6.2.9.7.2.5 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:ESFLatness:SDEViation
```

class StandardDevCls

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

class FetchStruct

Response structure. Fields:

- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in slots
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Ripple_1: float: float Max (range 1) - min (range 1) Unit: dB
- Ripple_2: float: float Max (range 2) - min (range 2) Unit: dB
- Max_R_1_Min_R_2: float: float Max (range 1) - min (range 2) Unit: dB
- Max_R_2_Min_R_1: float: float Max (range 2) - min (range 1) Unit: dB
- Min_R_1: float: float Min (range 1) Unit: dB
- Max_R_1: float: float Max (range 1) Unit: dB
- Min_R_2: float: float Min (range 2) Unit: dB
- Max_R_2: float: float Max (range 2) Unit: dB

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:ESFLatness:SDEViation
value: FetchStruct = driver.multiEval.listPy.segment.esFlatness.standardDev.
↪fetch(segment = repcap.Segment.Default)
```

Return equalizer spectrum flatness single value results for segment <no> in list mode.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

6.2.9.7.3 InbandEmission

class InbandEmissionCls

InbandEmission commands group definition. 18 total commands, 3 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.9.7.3.1 Cc<CarrierComponent>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.multiEval.listPy.segment.inbandEmission.cc.repcap_carrierComponent_get()
driver.multiEval.listPy.segment.inbandEmission.cc.repcap_carrierComponent_set(repcap.
↪CarrierComponent.Nr1)
```

class CcCls

Cc commands group definition. 6 total commands, 1 Subgroups, 0 group commands Repeated Capability: CarrierComponent, default value after init: CarrierComponent.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.segment.inbandEmission.cc.clone()
```

Subgroups

6.2.9.7.3.2 Margin

class MarginCls

Margin commands group definition. 6 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.segment.inbandEmission.cc.margin.clone()
```

Subgroups

6.2.9.7.3.3 Average

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:IEMission:CC<c>
↪:MARGin:AVERage
```

class AverageCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in slots
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Margin: float: float Unit: dB

fetch(*segment=Segment.Default*, *carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:IEMission:CC<c>:MARGin:AVERage
value: FetchStruct = driver.multiEval.listPy.segment.inbandEmission.cc.margin.
↪average.fetch(segment = repcap.Segment.Default, carrierComponent = repcap.
↪CarrierComponent.Default)
```

Return the inband emission limit line margin results for component carrier CC<c>, segment <no> in list mode. The CURRent margins indicate the minimum (vertical) distance between the limit line and the current trace. A negative result indicates that the limit is exceeded. The AVERage, EXTReme and SDEViation values are calculated from the current margins.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

6.2.9.7.3.4 Current

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:IEMission:CC<c>
↪:MARGin:CURRent
```

class CurrentCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in slots
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Margin: float: float Unit: dB

fetch(*segment=Segment.Default*, *carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:IEMission:CC<c>:MARGin:CURRent
value: FetchStruct = driver.multiEval.listPy.segment.inbandEmission.cc.margin.
↪current.fetch(segment = repcap.Segment.Default, carrierComponent = repcap.
↪CarrierComponent.Default)
```

Return the inband emission limit line margin results for component carrier CC<c>, segment <no> in list mode. The CURRent margins indicate the minimum (vertical) distance between the limit line and the current trace. A negative result indicates that the limit is exceeded. The AVERage, EXTReme and SDEViation values are calculated from the current margins.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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.multiEval.listPy.segment.inbandEmission.cc.margin.current.clone()
```

Subgroups

6.2.9.7.3.5 RbIndex

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:IEMission:CC<c>
↪:MARGin:CURRent:RBINdex
```

class RbIndexCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in slots
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Rb_Index: int: decimal Resource block index of margin

fetch(*segment=Segment.Default*, *carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:IEMission:CC<c>:MARGin:CURRent:RBINdex
value: FetchStruct = driver.multiEval.listPy.segment.inbandEmission.cc.margin.
↪current.rbIndex.fetch(segment = repcap.Segment.Default, carrierComponent =␣
↪repcap.CarrierComponent.Default)
```

Return resource block indices of the component carrier CC<c> inband emission measurement for segment <no> in list mode. At these RB indices, the CURRent and EXTReme margins have been detected.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

6.2.9.7.3.6 Extreme

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:IEMission:CC<c>
↪:MARGin:EXTReme
```

class ExtremeCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in slots
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Margin: float: float Unit: dB

fetch(*segment=Segment.Default*, *carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:IEMission:CC<c>:MARGin:EXTReme
value: FetchStruct = driver.multiEval.listPy.segment.inbandEmission.cc.margin.
↪extreme.fetch(segment = repcap.Segment.Default, carrierComponent = repcap.
↪CarrierComponent.Default)
```

Return the inband emission limit line margin results for component carrier CC<c>, segment <no> in list mode. The CURRent margins indicate the minimum (vertical) distance between the limit line and the current trace. A negative result indicates that the limit is exceeded. The AVERage, EXTReme and SDEViation values are calculated from the current margins.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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.multiEval.listPy.segment.inbandEmission.cc.margin.extreme.clone()
```

Subgroups

6.2.9.7.3.7 RbIndex

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:IEMission:CC<c>
↪:MARGin:EXTReme:RBINdex
```

class RbIndexCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in slots
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Rb_Index: int: decimal Resource block index of margin

fetch(*segment=Segment.Default*, *carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:IEMission:CC<c>:MARGin:EXTReme:RBINdex
value: FetchStruct = driver.multiEval.listPy.segment.inbandEmission.cc.margin.
↪extreme.rbIndex.fetch(segment = repcap.Segment.Default, carrierComponent =␣
↪repcap.CarrierComponent.Default)
```

Return resource block indices of the component carrier CC<c> inband emission measurement for segment <no> in list mode. At these RB indices, the CURRent and EXTReme margins have been detected.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

6.2.9.7.3.8 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:IEMission:CC<c>
↪:MARGin:SDEViation
```

class StandardDevCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in slots
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Margin: float: float Unit: dB

fetch(*segment=Segment.Default*, *carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:IEMission:CC<c>:MARGin:SDEViation
value: FetchStruct = driver.multiEval.listPy.segment.inbandEmission.cc.margin.
↪standardDev.fetch(segment = repcap.Segment.Default, carrierComponent = repcap.
↪CarrierComponent.Default)
```

Return the inband emission limit line margin results for component carrier CC<c>, segment <no> in list mode. The CURRent margins indicate the minimum (vertical) distance between the limit line and the current trace. A negative result indicates that the limit is exceeded. The AVERage, EXTReme and SDEViation values are calculated from the current margins.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

6.2.9.7.3.9 Margin

class MarginCls

Margin commands group definition. 6 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.segment.inbandEmission.margin.clone()
```

Subgroups

6.2.9.7.3.10 Average

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:IEMission:MARGin:AVERage
```

class AverageCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Margin: float: No parameter help available

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:IEMission:MARGin:AVERage
value: FetchStruct = driver.multiEval.listPy.segment.inbandEmission.margin.
↪average.fetch(segment = repcap.Segment.Default)
```

No command help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

6.2.9.7.3.11 Current

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:IEMission:MARGin:CURRent
```

class CurrentCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available

- Out_Of_Tolerance: int: No parameter help available
- Margin: float: No parameter help available

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:IEMission:MARGin:CURRent
value: FetchStruct = driver.multiEval.listPy.segment.inbandEmission.margin.
↪current.fetch(segment = repcap.Segment.Default)
```

No command help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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.multiEval.listPy.segment.inbandEmission.margin.current.clone()
```

Subgroups

6.2.9.7.3.12 RbIndex

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:IEMission:MARGin:CURRent:RBINdex
```

class RbIndexCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Rb_Index: int: No parameter help available

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:IEMission:MARGin:CURRent:RBINdex
value: FetchStruct = driver.multiEval.listPy.segment.inbandEmission.margin.
↪current.rbIndex.fetch(segment = repcap.Segment.Default)
```

No command help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

6.2.9.7.3.13 Extreme

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:IEMission:MARGin:EXTReme
```

class ExtremeCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Margin: float: No parameter help available

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:IEMission:MARGin:EXTReme
value: FetchStruct = driver.multiEval.listPy.segment.inbandEmission.margin.
↪extreme.fetch(segment = repcap.Segment.Default)
```

No command help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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.multiEval.listPy.segment.inbandEmission.margin.extreme.clone()
```

Subgroups

6.2.9.7.3.14 RbIndex

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:IEMission:MARGin:EXTReme:RBINdex
```

class RbIndexCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Rb_Index: int: No parameter help available

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:IEMission:MARGin:EXTReme:RBINdex
value: FetchStruct = driver.multiEval.listPy.segment.inbandEmission.margin.
↪extreme.rbIndex.fetch(segment = repcap.Segment.Default)
```

No command help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

6.2.9.7.3.15 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:IEMission:MARGin:SDEViation
```

class StandardDevCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available

- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Margin: float: No parameter help available

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:IEMission:MARGin:SDEViation
value: FetchStruct = driver.multiEval.listPy.segment.inbandEmission.margin.
↪standardDev.fetch(segment = repcap.Segment.Default)
```

No command help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

6.2.9.7.3.16 Scc<SecondaryCC>

RepCap Settings

```
# Range: CC1 .. CC7
rc = driver.multiEval.listPy.segment.inbandEmission.scc.repcap_secondaryCC_get()
driver.multiEval.listPy.segment.inbandEmission.scc.repcap_secondaryCC_set(repcap.
↪SecondaryCC.CC1)
```

class SccCls

Scc commands group definition. 6 total commands, 1 Subgroups, 0 group commands Repeated Capability: SecondaryCC, default value after init: SecondaryCC.CC1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.segment.inbandEmission.scc.clone()
```

Subgroups

6.2.9.7.3.17 Margin

class MarginCls

Margin commands group definition. 6 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.segment.inbandEmission.scc.margin.clone()
```

Subgroups

6.2.9.7.3.18 Average

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:IEMission:SCC<c>
↪:MARGin:AVERage
```

class AverageCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Margin: float: No parameter help available

fetch(*segment=Segment.Default*, *secondaryCC=SecondaryCC.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:IEMission:SCC<c>:MARGin:AVERage
value: FetchStruct = driver.multiEval.listPy.segment.inbandEmission.scc.margin.
↪average.fetch(segment = repcap.Segment.Default, secondaryCC = repcap.
↪SecondaryCC.Default)
```

No command help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

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

6.2.9.7.3.19 Current

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:IEMission:SCC<c>
↪:MARGin:CURRent
```

class CurrentCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Margin: float: No parameter help available

fetch(*segment=Segment.Default*, *secondaryCC=SecondaryCC.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:IEMission:SCC<c>:MARGin:CURRent
value: FetchStruct = driver.multiEval.listPy.segment.inbandEmission.scc.margin.
↪current.fetch(segment = repcap.Segment.Default, secondaryCC = repcap.
↪SecondaryCC.Default)
```

No command help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

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.multiEval.listPy.segment.inbandEmission.scc.margin.current.clone()
```

Subgroups

6.2.9.7.3.20 RbIndex

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:IEMission:SCC<c>
↪:MARGin:CURRent:RBINdex
```

class RbIndexCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Rb_Index: int: No parameter help available

fetch(*segment=Segment.Default*, *secondaryCC=SecondaryCC.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:IEMission:SCC<c>:MARGin:CURRent:RBINdex
value: FetchStruct = driver.multiEval.listPy.segment.inbandEmission.scc.margin.
↪current.rbIndex.fetch(segment = repcap.Segment.Default, secondaryCC = repcap.
↪SecondaryCC.Default)
```

No command help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

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

6.2.9.7.3.21 Extreme

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:IEMission:SCC<c>
↪:MARGin:EXTReme
```

class ExtremeCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Margin: float: No parameter help available

fetch(*segment=Segment.Default*, *secondaryCC=SecondaryCC.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:IEMission:SCC<c>:MARGin:EXTReme
value: FetchStruct = driver.multiEval.listPy.segment.inbandEmission.scc.margin.
↪extreme.fetch(segment = repcap.Segment.Default, secondaryCC = repcap.
↪SecondaryCC.Default)
```

No command help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

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.multiEval.listPy.segment.inbandEmission.scc.margin.extreme.clone()
```

Subgroups

6.2.9.7.3.22 RbIndex

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:IEMission:SCC<c>
↪:MARGin:EXTReme:RBINdex
```

class RbIndexCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available

- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Rb_Index: int: No parameter help available

fetch(*segment=Segment.Default*, *secondaryCC=SecondaryCC.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:IEMission:SCC<c>:MARGin:EXTReme:RBINdex
value: FetchStruct = driver.multiEval.listPy.segment.inbandEmission.scc.margin.
↪extreme.rbIndex.fetch(segment = repcap.Segment.Default, secondaryCC = repcap.
↪SecondaryCC.Default)
```

No command help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

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

6.2.9.7.3.23 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:IEMission:SCC<c>
↪:MARGin:SDEViation
```

class StandardDevCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Margin: float: No parameter help available

fetch(*segment=Segment.Default*, *secondaryCC=SecondaryCC.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:IEMission:SCC<c>:MARGin:SDEViation
value: FetchStruct = driver.multiEval.listPy.segment.inbandEmission.scc.margin.
↪standardDev.fetch(segment = repcap.Segment.Default, secondaryCC = repcap.
↪SecondaryCC.Default)
```

No command help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

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

6.2.9.7.4 Modulation

class ModulationCls

Modulation commands group definition. 11 total commands, 8 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.9.7.4.1 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:MODulation:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:MODulation:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in slots
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Evm_Rms_Low: float or bool: float EVM RMS value, low EVM window position Unit: %
- Evm_Rms_High: float or bool: float EVM RMS value, high EVM window position Unit: %
- Evm_Peak_Low: float or bool: float EVM peak value, low EVM window position Unit: %
- Evm_Peak_High: float or bool: float EVM peak value, high EVM window position Unit: %
- Mag_Error_Rms_Low: float or bool: float Magnitude error RMS value, low EVM window position Unit: %

- Mag_Error_Rms_High: float or bool: float Magnitude error RMS value, low EVM window position Unit: %
- Mag_Error_Peak_Low: float or bool: float Magnitude error peak value, low EVM window position Unit: %
- Mag_Err_Peak_High: float or bool: float Magnitude error peak value, high EVM window position Unit: %
- Ph_Error_Rms_Low: float or bool: float Phase error RMS value, low EVM window position Unit: deg
- Ph_Error_Rms_High: float or bool: float Phase error RMS value, high EVM window position Unit: deg
- Ph_Error_Peak_Low: float or bool: float Phase error peak value, low EVM window position Unit: deg
- Ph_Error_Peak_High: float or bool: float Phase error peak value, high EVM window position Unit: deg
- Iq_Offset: float or bool: float I/Q origin offset Unit: dBc
- Frequency_Error: float or bool: float Carrier frequency error Unit: Hz
- Timing_Error: float or bool: float Time error Unit: Ts (basic LTE time unit)
- Tx_Power: float or bool: float User equipment power Unit: dBm
- Peak_Power: float or bool: float User equipment peak power Unit: dBm
- Psd: float or bool: No parameter help available
- Evm_Dmrs_Low: float or bool: float EVM DMRS value, low EVM window position Unit: %
- Evm_Dmrs_High: float or bool: float EVM DMRS value, high EVM window position Unit: %
- Mag_Err_Dmrs_Low: float or bool: float Magnitude error DMRS value, low EVM window position Unit: %
- Mag_Err_Dmrs_High: float or bool: float Magnitude error DMRS value, high EVM window position Unit: %
- Ph_Error_Dmrs_Low: float or bool: float Phase error DMRS value, low EVM window position Unit: deg
- Ph_Error_Dmrs_High: float or bool: float Phase error DMRS value, high EVM window position Unit: deg

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in slots
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Evm_Rms_Low: float: float EVM RMS value, low EVM window position Unit: %
- Evm_Rms_High: float: float EVM RMS value, high EVM window position Unit: %
- Evm_Peak_Low: float: float EVM peak value, low EVM window position Unit: %
- Evm_Peak_High: float: float EVM peak value, high EVM window position Unit: %
- Mag_Error_Rms_Low: float: float Magnitude error RMS value, low EVM window position Unit: %

- Mag_Error_Rms_High: float: float Magnitude error RMS value, low EVM window position Unit: %
- Mag_Error_Peak_Low: float: float Magnitude error peak value, low EVM window position Unit: %
- Mag_Err_Peak_High: float: float Magnitude error peak value, high EVM window position Unit: %
- Ph_Error_Rms_Low: float: float Phase error RMS value, low EVM window position Unit: deg
- Ph_Error_Rms_High: float: float Phase error RMS value, high EVM window position Unit: deg
- Ph_Error_Peak_Low: float: float Phase error peak value, low EVM window position Unit: deg
- Ph_Error_Peak_High: float: float Phase error peak value, high EVM window position Unit: deg
- Iq_Offset: float: float I/Q origin offset Unit: dBc
- Frequency_Error: float: float Carrier frequency error Unit: Hz
- Timing_Error: float: float Time error Unit: Ts (basic LTE time unit)
- Tx_Power: float: float User equipment power Unit: dBm
- Peak_Power: float: float User equipment peak power Unit: dBm
- Psd: float: No parameter help available
- Evm_Dmrs_Low: float: float EVM DMRS value, low EVM window position Unit: %
- Evm_Dmrs_High: float: float EVM DMRS value, high EVM window position Unit: %
- Mag_Err_Dmrs_Low: float: float Magnitude error DMRS value, low EVM window position Unit: %
- Mag_Err_Dmrs_High: float: float Magnitude error DMRS value, high EVM window position Unit: %
- Ph_Error_Dmrs_Low: float: float Phase error DMRS value, low EVM window position Unit: deg
- Ph_Error_Dmrs_High: float: float Phase error DMRS value, high EVM window position Unit: deg

calculate(*segment=Segment.Default*) → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:MODulation:AVERage
value: CalculateStruct = driver.multiEval.listPy.segment.modulation.average.
↪calculate(segment = repcap.Segment.Default)
```

Returns modulation single-value results for segment <no> in list mode. The values described below are returned by FETCh commands. The first four values (reliability to out-of-tolerance result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:MODulation:AVERage
value: FetchStruct = driver.multiEval.listPy.segment.modulation.average.
↪fetch(segment = repcap.Segment.Default)
```

Returns modulation single-value results for segment <no> in list mode. The values described below are returned by FETCh commands. The first four values (reliability to out-of-tolerance result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

6.2.9.7.4.2 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:MODulation:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:MODulation:CURRent
```

`class CurrentCls`

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

`class CalculateStruct`

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in slots
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Evm_Rms_Low: float or bool: float EVM RMS value, low EVM window position Unit: %
- Evm_Rms_High: float or bool: float EVM RMS value, high EVM window position Unit: %
- Evm_Peak_Low: float or bool: float EVM peak value, low EVM window position Unit: %
- Evm_Peak_High: float or bool: float EVM peak value, high EVM window position Unit: %
- Mag_Error_Rms_Low: float or bool: float Magnitude error RMS value, low EVM window position Unit: %
- Mag_Error_Rms_High: float or bool: float Magnitude error RMS value, low EVM window position Unit: %
- Mag_Error_Peak_Low: float or bool: float Magnitude error peak value, low EVM window position Unit: %
- Mag_Err_Peak_High: float or bool: float Magnitude error peak value, high EVM window position Unit: %
- Ph_Error_Rms_Low: float or bool: float Phase error RMS value, low EVM window position Unit: deg
- Ph_Error_Rms_High: float or bool: float Phase error RMS value, high EVM window position Unit: deg
- Ph_Error_Peak_Low: float or bool: float Phase error peak value, low EVM window position Unit: deg

- Ph_Error_Peak_High: float or bool: float Phase error peak value, high EVM window position Unit: deg
- Iq_Offset: float or bool: float I/Q origin offset Unit: dBc
- Frequency_Error: float or bool: float Carrier frequency error Unit: Hz
- Timing_Error: float or bool: float Time error Unit: Ts (basic LTE time unit)
- Tx_Power: float or bool: float User equipment power Unit: dBm
- Peak_Power: float or bool: float User equipment peak power Unit: dBm
- Psd: float or bool: No parameter help available
- Evm_Dmrs_Low: float or bool: float EVM DMRS value, low EVM window position Unit: %
- Evm_Dmrs_High: float or bool: float EVM DMRS value, high EVM window position Unit: %
- Mag_Err_Dmrs_Low: float or bool: float Magnitude error DMRS value, low EVM window position Unit: %
- Mag_Err_Dmrs_High: float or bool: float Magnitude error DMRS value, high EVM window position Unit: %
- Ph_Error_Dmrs_Low: float or bool: float Phase error DMRS value, low EVM window position Unit: deg
- Ph_Error_Dmrs_High: float or bool: float Phase error DMRS value, high EVM window position Unit: deg

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in slots
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Evm_Rms_Low: float: float EVM RMS value, low EVM window position Unit: %
- Evm_Rms_High: float: float EVM RMS value, high EVM window position Unit: %
- Evm_Peak_Low: float: float EVM peak value, low EVM window position Unit: %
- Evm_Peak_High: float: float EVM peak value, high EVM window position Unit: %
- Mag_Error_Rms_Low: float: float Magnitude error RMS value, low EVM window position Unit: %
- Mag_Error_Rms_High: float: float Magnitude error RMS value, low EVM window position Unit: %
- Mag_Error_Peak_Low: float: float Magnitude error peak value, low EVM window position Unit: %
- Mag_Err_Peak_High: float: float Magnitude error peak value, high EVM window position Unit: %
- Ph_Error_Rms_Low: float: float Phase error RMS value, low EVM window position Unit: deg
- Ph_Error_Rms_High: float: float Phase error RMS value, high EVM window position Unit: deg
- Ph_Error_Peak_Low: float: float Phase error peak value, low EVM window position Unit: deg
- Ph_Error_Peak_High: float: float Phase error peak value, high EVM window position Unit: deg
- Iq_Offset: float: float I/Q origin offset Unit: dBc
- Frequency_Error: float: float Carrier frequency error Unit: Hz

- Timing_Error: float: float Time error Unit: Ts (basic LTE time unit)
- Tx_Power: float: float User equipment power Unit: dBm
- Peak_Power: float: float User equipment peak power Unit: dBm
- Psd: float: No parameter help available
- Evm_Dmrs_Low: float: float EVM DMRS value, low EVM window position Unit: %
- Evm_Dmrs_High: float: float EVM DMRS value, high EVM window position Unit: %
- Mag_Err_Dmrs_Low: float: float Magnitude error DMRS value, low EVM window position Unit: %
- Mag_Err_Dmrs_High: float: float Magnitude error DMRS value, high EVM window position Unit: %
- Ph_Error_Dmrs_Low: float: float Phase error DMRS value, low EVM window position Unit: deg
- Ph_Error_Dmrs_High: float: float Phase error DMRS value, high EVM window position Unit: deg

calculate(*segment=Segment.Default*) → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:MODulation:CURRent
value: CalculateStruct = driver.multiEval.listPy.segment.modulation.current.
↪calculate(segment = repcap.Segment.Default)
```

Returns modulation single-value results for segment <no> in list mode. The values described below are returned by FETCh commands. The first four values (reliability to out-of-tolerance result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:MODulation:CURRent
value: FetchStruct = driver.multiEval.listPy.segment.modulation.current.
↪fetch(segment = repcap.Segment.Default)
```

Returns modulation single-value results for segment <no> in list mode. The values described below are returned by FETCh commands. The first four values (reliability to out-of-tolerance result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

6.2.9.7.4.3 Dallocation

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:MODulation:DALLocation
```

class DallocationCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Nr_Res_Blocks: int: decimal Number of allocated resource blocks
- Offset_Res_Blocks: int: decimal Offset of the first allocated resource block from the edge of the allocated UL transmission bandwidth

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:MODulation:DALLocation
value: FetchStruct = driver.multiEval.listPy.segment.modulation.dallocation.
↪fetch(segment = repcap.Segment.Default)
```

Return the detected allocation for segment <no> in list mode. The result is determined from the last measured slot of the statistical length. The individual measurements provide the same result when measuring the same slot. However different statistical lengths can be defined for the measurements so that the measured slots and returned results can differ.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

6.2.9.7.4.4 DchType

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:MODulation:DCHType
```

class DchTypeCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Channel_Type: enums.UplinkChannelType: PUSCh | PUCCh

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:MODulation:DCHType
value: FetchStruct = driver.multiEval.listPy.segment.modulation.dchType.
↪fetch(segment = repcap.Segment.Default)
```

Return the uplink channel type for segment <no> in list mode. The result is determined from the last measured slot of the statistical length. The individual measurements provide the same result when measuring the same slot. However different statistical lengths can be defined for the measurements so that the measured slots and returned results can differ.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

6.2.9.7.4.5 Dmodulation

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:MODulation:DMODulation
```

class DmodulationCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Modulation: enums.Modulation: QPSK | Q16 | Q64 | Q256 QPSK, 16-QAM, 64-QAM, 256-QAM

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:MODulation:DMODulation
value: FetchStruct = driver.multiEval.listPy.segment.modulation.dmodulation.
↪fetch(segment = repcap.Segment.Default)
```

Return the detected modulation scheme for segment <no> in list mode. The result is determined from the last measured slot of the statistical length. If channel type PUCCH is detected, QPSK is returned as modulation type because the QPSK limits are applied in that case.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

6.2.9.7.4.6 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:MODulation:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:MODulation:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in slots
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Evm_Rms_Low: float or bool: float EVM RMS value, low EVM window position Unit: %
- Evm_Rms_High: float or bool: float EVM RMS value, high EVM window position Unit: %
- Evm_Peak_Low: float or bool: float EVM peak value, low EVM window position Unit: %
- Evm_Peak_High: float or bool: float EVM peak value, high EVM window position Unit: %
- Mag_Error_Rms_Low: float or bool: float Magnitude error RMS value, low EVM window position Unit: %
- Mag_Error_Rms_High: float or bool: float Magnitude error RMS value, low EVM window position Unit: %
- Mag_Error_Peak_Low: float or bool: float Magnitude error peak value, low EVM window position Unit: %
- Mag_Err_Peak_High: float or bool: float Magnitude error peak value, high EVM window position Unit: %
- Ph_Error_Rms_Low: float or bool: float Phase error RMS value, low EVM window position Unit: deg
- Ph_Error_Rms_High: float or bool: float Phase error RMS value, high EVM window position Unit: deg
- Ph_Error_Peak_Low: float or bool: float Phase error peak value, low EVM window position Unit: deg
- Ph_Error_Peak_High: float or bool: float Phase error peak value, high EVM window position Unit: deg
- Iq_Offset: float or bool: float I/Q origin offset Unit: dBc
- Frequency_Error: float or bool: float Carrier frequency error Unit: Hz
- Timing_Error: float or bool: float Time error Unit: Ts (basic LTE time unit)
- Tx_Power_Minimum: float or bool: float Minimum user equipment power Unit: dBm
- Tx_Power_Maximum: float or bool: float Maximum user equipment power Unit: dBm
- Peak_Power_Min: float or bool: float Minimum user equipment peak power Unit: dBm
- Peak_Power_Max: float or bool: float Maximum user equipment peak power Unit: dBm

- Psd_Minimum: float or bool: No parameter help available
- Psd_Maximum: float or bool: No parameter help available
- Evm_Dmrs_Low: float or bool: float EVM DMRS value, low EVM window position Unit: %
- Evm_Dmrs_High: float or bool: float EVM DMRS value, high EVM window position Unit: %
- Mag_Err_Dmrs_Low: float or bool: float Magnitude error DMRS value, low EVM window position Unit: %
- Mag_Err_Dmrs_High: float or bool: float Magnitude error DMRS value, high EVM window position Unit: %
- Ph_Error_Dmrs_Low: float or bool: float Phase error DMRS value, low EVM window position Unit: deg
- Ph_Error_Dmrs_High: float or bool: float Phase error DMRS value, high EVM window position Unit: deg

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in slots
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Evm_Rms_Low: float: float EVM RMS value, low EVM window position Unit: %
- Evm_Rms_High: float: float EVM RMS value, high EVM window position Unit: %
- Evm_Peak_Low: float: float EVM peak value, low EVM window position Unit: %
- Evm_Peak_High: float: float EVM peak value, high EVM window position Unit: %
- Mag_Error_Rms_Low: float: float Magnitude error RMS value, low EVM window position Unit: %
- Mag_Error_Rms_High: float: float Magnitude error RMS value, low EVM window position Unit: %
- Mag_Error_Peak_Low: float: float Magnitude error peak value, low EVM window position Unit: %
- Mag_Err_Peak_High: float: float Magnitude error peak value, high EVM window position Unit: %
- Ph_Error_Rms_Low: float: float Phase error RMS value, low EVM window position Unit: deg
- Ph_Error_Rms_High: float: float Phase error RMS value, high EVM window position Unit: deg
- Ph_Error_Peak_Low: float: float Phase error peak value, low EVM window position Unit: deg
- Ph_Error_Peak_High: float: float Phase error peak value, high EVM window position Unit: deg
- Iq_Offset: float: float I/Q origin offset Unit: dBc
- Frequency_Error: float: float Carrier frequency error Unit: Hz
- Timing_Error: float: float Time error Unit: Ts (basic LTE time unit)
- Tx_Power_Minimum: float: float Minimum user equipment power Unit: dBm
- Tx_Power_Maximum: float: float Maximum user equipment power Unit: dBm
- Peak_Power_Min: float: float Minimum user equipment peak power Unit: dBm
- Peak_Power_Max: float: float Maximum user equipment peak power Unit: dBm
- Psd_Minimum: float: No parameter help available

- Psd_Maximum: float: No parameter help available
- Evm_Dmrs_Low: float: float EVM DMRS value, low EVM window position Unit: %
- Evm_Dmrs_High: float: float EVM DMRS value, high EVM window position Unit: %
- Mag_Err_Dmrs_Low: float: float Magnitude error DMRS value, low EVM window position Unit: %
- Mag_Err_Dmrs_High: float: float Magnitude error DMRS value, high EVM window position Unit: %
- Ph_Error_Dmrs_Low: float: float Phase error DMRS value, low EVM window position Unit: deg
- Ph_Error_Dmrs_High: float: float Phase error DMRS value, high EVM window position Unit: deg

calculate(*segment=Segment.Default*) → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:MODulation:EXTReme
value: CalculateStruct = driver.multiEval.listPy.segment.modulation.extreme.
↪calculate(segment = repcap.Segment.Default)
```

Return modulation single-value results for segment <no> in list mode. The values described below are returned by FETCh commands. The first four values (reliability to out-of-tolerance result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:MODulation:EXTReme
value: FetchStruct = driver.multiEval.listPy.segment.modulation.extreme.
↪fetch(segment = repcap.Segment.Default)
```

Return modulation single-value results for segment <no> in list mode. The values described below are returned by FETCh commands. The first four values (reliability to out-of-tolerance result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

6.2.9.7.4.7 SchType

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:MODulation:SCHType
```

class SchTypeCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Channel_Type: enums.SidelinkChannelType: PSSCh | PSCCh

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:MODulation:SCHType
value: FetchStruct = driver.multiEval.listPy.segment.modulation.schType.
↪fetch(segment = repcap.Segment.Default)
```

Returns the sidelink channel type evaluated for modulation results, for segment <no> in list mode.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

6.2.9.7.4.8 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:MODulation:SDEViation
```

class StandardDevCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in slots
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Evm_Rms_Low: float: float EVM RMS value, low EVM window position Unit: %
- Evm_Rms_High: float: float EVM RMS value, high EVM window position Unit: %

- Evm_Peak_Low: float: float EVM peak value, low EVM window position Unit: %
- Evm_Peak_High: float: float EVM peak value, high EVM window position Unit: %
- Mag_Error_Rms_Low: float: float Magnitude error RMS value, low EVM window position Unit: %
- Mag_Error_Rms_High: float: float Magnitude error RMS value, low EVM window position Unit: %
- Mag_Error_Peak_Low: float: float Magnitude error peak value, low EVM window position Unit: %
- Mag_Err_Peak_High: float: float Magnitude error peak value, high EVM window position Unit: %
- Ph_Error_Rms_Low: float: float Phase error RMS value, low EVM window position Unit: deg
- Ph_Error_Rms_High: float: float Phase error RMS value, high EVM window position Unit: deg
- Ph_Error_Peak_Low: float: float Phase error peak value, low EVM window position Unit: deg
- Ph_Error_Peak_High: float: float Phase error peak value, high EVM window position Unit: deg
- Iq_Offset: float: float I/Q origin offset Unit: dBc
- Frequency_Error: float: float Carrier frequency error Unit: Hz
- Timing_Error: float: float Time error Unit: Ts (basic LTE time unit)
- Tx_Power: float: float User equipment power Unit: dBm
- Peak_Power: float: float User equipment peak power Unit: dBm
- Psd: float: No parameter help available
- Evm_Dmrs_Low: float: float EVM DMRS value, low EVM window position Unit: %
- Evm_Dmrs_High: float: float EVM DMRS value, high EVM window position Unit: %
- Mag_Err_Dmrs_Low: float: float Magnitude error DMRS value, low EVM window position Unit: %
- Mag_Err_Dmrs_High: float: float Magnitude error DMRS value, high EVM window position Unit: %
- Ph_Error_Dmrs_Low: float: float Phase error DMRS value, low EVM window position Unit: deg
- Ph_Error_Dmrs_High: float: float Phase error DMRS value, high EVM window position Unit: deg

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:MODulation:SDEViation
value: FetchStruct = driver.multiEval.listPy.segment.modulation.standardDev.
↪fetch(segment = repcap.Segment.Default)
```

Returns modulation single-value results for segment <no> in list mode. The values described below are returned by FETCh commands. The first four values (reliability to out-of-tolerance result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

6.2.9.7.5 Pmonitor

class PmonitorCls

Pmonitor commands group definition. 4 total commands, 3 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.9.7.5.1 Array

class ArrayCls

Array 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.multiEval.listPy.segment.pmonitor.array.clone()
```

Subgroups

6.2.9.7.5.2 Length

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:PMONitor:ARRay:LENGth
```

class LengthCls

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

fetch(*segment=Segment.Default*) → int

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:PMONitor:ARRay:LENGth
value: int = driver.multiEval.listPy.segment.pmonitor.array.length.
↪fetch(segment = repcap.Segment.Default)
```

Returns the number of power monitor results for segment <no> contained in a result list for all measured segments. Such a result list is, for example, returned by the command method RsCmwLteMeas.MultiEval.ListPy.Pmonitor.Rms.fetch.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
length: decimal Number of power monitor results

6.2.9.7.5.3 Start

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:PMONitor:ARRay:STARt
```

class StartCls

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

fetch(*segment=Segment.Default*) → int

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:PMONitor:ARRay:STARt
value: int = driver.multiEval.listPy.segment.pmonitor.array.start.fetch(segment␣
↪= repcap.Segment.Default)
```

Returns the offset of the first power monitor result for segment <no> within a result list for all measured segments. Such a result list is, for example, returned by the command method RsCmwLteMeas.MultiEval.ListPy.Pmonitor.Rms.fetch. A returned <Start> value n indicates that the result for the first subframe of the segment is the (n+1) th result in the power result list over all segments.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
start: decimal Offset of the first power monitor result

6.2.9.7.5.4 Peak

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:PMONitor:PEAK
```

class PeakCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Step_Peak_Power: List[float]: No parameter help available

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:PMONitor:PEAK
value: FetchStruct = driver.multiEval.listPy.segment.pmonitor.peak.
↪fetch(segment = repcap.Segment.Default)
```

Return the power monitor results for segment <no> in list mode. The commands return one power result for each subframe of the segment for the measured carrier. The power values are RMS averaged over the subframe or represent the peak value within the subframe.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

6.2.9.7.5.5 Rms

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:PMONitor:RMS
```

class RmsCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Step_Rms_Power: List[float]: No parameter help available

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:PMONitor:RMS
value: FetchStruct = driver.multiEval.listPy.segment.pmonitor.rms.fetch(segment␣
↪= repcap.Segment.Default)
```

Return the power monitor results for segment <no> in list mode. The commands return one power result for each subframe of the segment for the measured carrier. The power values are RMS averaged over the subframe or represent the peak value within the subframe.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

6.2.9.7.6 Power

class PowerCls

Power 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.multiEval.listPy.segment.power.clone()
```

Subgroups

6.2.9.7.6.1 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Tx_Power: float or bool: No parameter help available

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in subframes
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Tx_Power: float: float Total TX power of all component carriers Unit: dBm

calculate(*segment=Segment.Default*) → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:POWer:AVERage
value: CalculateStruct = driver.multiEval.listPy.segment.power.average.
↪calculate(segment = repcap.Segment.Default)
```

No command help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:POWer:AVERage
value: FetchStruct = driver.multiEval.listPy.segment.power.average.
↪fetch(segment = repcap.Segment.Default)
```

Return total TX power results for segment <no> in list mode.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

6.2.9.7.6.2 Cc<CarrierComponent>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.multiEval.listPy.segment.power.cc.repcap_carrierComponent_get()
driver.multiEval.listPy.segment.power.cc.repcap_carrierComponent_set(repcap.
↪CarrierComponent.Nr1)
```

class CcCls

Cc commands group definition. 9 total commands, 5 Subgroups, 0 group commands Repeated Capability: CarrierComponent, default value after init: CarrierComponent.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.segment.power.cc.clone()
```

Subgroups

6.2.9.7.6.3 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:CC<no>:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:CC<no>:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Tx_Power: float or bool: No parameter help available

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in subframes
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Tx_Power: float: float TX power of the component carrier Unit: dBm

calculate(*segment=Segment.Default*, *carrierComponent=CarrierComponent.Default*) → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:POWer:CC<no>:AVERage
value: CalculateStruct = driver.multiEval.listPy.segment.power.cc.average.
↪calculate(segment = repcap.Segment.Default, carrierComponent = repcap.
↪CarrierComponent.Default)
```

No command help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

fetch(*segment=Segment.Default*, *carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:CC
↪<no>:AVERage
value: FetchStruct = driver.multiEval.listPy.segment.power.cc.average.
↪fetch(segment = repcap.Segment.Default, carrierComponent = repcap.
↪CarrierComponent.Default)
```

Return TX power results for component carrier CC<no> and a single segment in list mode.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

6.2.9.7.6.4 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:CC<no>:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:CC<no>:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Tx_Power: float or bool: No parameter help available

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in subframes
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Tx_Power: float: float TX power of the component carrier Unit: dBm

calculate(*segment=Segment.Default*, *carrierComponent=CarrierComponent.Default*) → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:POWer:CC<no>:CURRent
value: CalculateStruct = driver.multiEval.listPy.segment.power.cc.current.
↪calculate(segment = repcap.Segment.Default, carrierComponent = repcap.
↪CarrierComponent.Default)
```

No command help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

fetch(*segment=Segment.Default*, *carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:CC
↪<no>:CURRent
value: FetchStruct = driver.multiEval.listPy.segment.power.cc.current.
↪fetch(segment = repcap.Segment.Default, carrierComponent = repcap.
↪CarrierComponent.Default)
```

Return TX power results for component carrier CC<no> and a single segment in list mode.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

6.2.9.7.6.5 Maximum

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:CC<no>:MAXimum
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:CC<no>:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Tx_Power_Max: float or bool: No parameter help available

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in subframes
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Tx_Power_Max: float: No parameter help available

calculate(*segment=Segment.Default*, *carrierComponent=CarrierComponent.Default*) → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:POWer:CC<no>:MAXimum
value: CalculateStruct = driver.multiEval.listPy.segment.power.cc.maximum.
↪calculate(segment = repcap.Segment.Default, carrierComponent = repcap.
↪CarrierComponent.Default)
```

No command help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

fetch(*segment=Segment.Default*, *carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:CC
↪<no>:MAXimum
value: FetchStruct = driver.multiEval.listPy.segment.power.cc.maximum.
↪fetch(segment = repcap.Segment.Default, carrierComponent = repcap.
↪CarrierComponent.Default)
```

Return TX power results for component carrier CC<no> and a single segment in list mode.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

6.2.9.7.6.6 Minimum

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:CC<no>:MINimum
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:CC<no>:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available

- Tx_Power_Min: float or bool: No parameter help available

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in subframes
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Tx_Power_Min: float: No parameter help available

calculate(*segment=Segment.Default*, *carrierComponent=CarrierComponent.Default*) → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:POWer:CC<no>:MINimum
value: CalculateStruct = driver.multiEval.listPy.segment.power.cc.minimum.
↪calculate(segment = repcap.Segment.Default, carrierComponent = repcap.
↪CarrierComponent.Default)
```

No command help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

fetch(*segment=Segment.Default*, *carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:CC
↪<no>:MINimum
value: FetchStruct = driver.multiEval.listPy.segment.power.cc.minimum.
↪fetch(segment = repcap.Segment.Default, carrierComponent = repcap.
↪CarrierComponent.Default)
```

Return TX power results for component carrier CC<no> and a single segment in list mode.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

6.2.9.7.6.7 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:CC<no>:SDEViation
```

class StandardDevCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in subframes
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Tx_Power: float: float TX power of the component carrier Unit: dBm

fetch(*segment=Segment.Default*, *carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:CC
↪<no>:SDEViation
value: FetchStruct = driver.multiEval.listPy.segment.power.cc.standardDev.
↪fetch(segment = repcap.Segment.Default, carrierComponent = repcap.
↪CarrierComponent.Default)
```

Return TX power results for component carrier CC<no> and a single segment in list mode.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

6.2.9.7.6.8 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available

- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Tx_Power: float or bool: No parameter help available

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in subframes
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Tx_Power: float: float Total TX power of all component carriers Unit: dBm

calculate(*segment=Segment.Default*) → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:POWer:CURRent
value: CalculateStruct = driver.multiEval.listPy.segment.power.current.
↪calculate(segment = repcap.Segment.Default)
```

No command help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:POWer:CURRent
value: FetchStruct = driver.multiEval.listPy.segment.power.current.
↪fetch(segment = repcap.Segment.Default)
```

Return total TX power results for segment <no> in list mode.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

6.2.9.7.6.9 Maximum

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:MAXimum
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Tx_Power_Max: float or bool: No parameter help available

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in subframes
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Tx_Power_Max: float: No parameter help available

calculate(*segment=Segment.Default*) → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:POWer:MAXimum
value: CalculateStruct = driver.multiEval.listPy.segment.power.maximum.
↪calculate(segment = repcap.Segment.Default)
```

No command help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:POWer:MAXimum
value: FetchStruct = driver.multiEval.listPy.segment.power.maximum.
↪fetch(segment = repcap.Segment.Default)
```

Return total TX power results for segment <no> in list mode.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

6.2.9.7.6.10 Minimum

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:MINimum
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Tx_Power_Min: float or bool: No parameter help available

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in subframes
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Tx_Power_Min: float: No parameter help available

calculate(*segment=Segment.Default*) → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:POWer:MINimum
value: CalculateStruct = driver.multiEval.listPy.segment.power.minimum.
↪calculate(segment = repcap.Segment.Default)
```

No command help available

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:POWer:MINimum
value: FetchStruct = driver.multiEval.listPy.segment.power.minimum.
↪fetch(segment = repcap.Segment.Default)
```

Return total TX power results for segment <no> in list mode.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

6.2.9.7.6.11 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:SDEViation
```

class StandardDevCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in subframes
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Tx_Power: float: float Total TX power of all component carriers Unit: dBm

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:POWer:SDEViation
value: FetchStruct = driver.multiEval.listPy.segment.power.standardDev.
↪fetch(segment = repcap.Segment.Default)
```

Return total TX power results for segment <no> in list mode.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

6.2.9.7.7 SeMask

class SeMaskCls

SeMask commands group definition. 16 total commands, 7 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.9.7.7.1 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SEMask:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SEMask:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in slots
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Obw: float or bool: float Occupied bandwidth Unit: Hz
- Tx_Power: float or bool: float Total TX power in the slot Unit: dBm

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in slots
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Obw: float: float Occupied bandwidth Unit: Hz
- Tx_Power: float: float Total TX power in the slot Unit: dBm

calculate(*segment=Segment.Default*) → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:AVERage
value: CalculateStruct = driver.multiEval.listPy.segment.seMask.average.
↪calculate(segment = repcap.Segment.Default)
```

Return spectrum emission single value results for segment <no> in list mode. The values described below are returned by FETCh commands. The first four values (reliability to out-of-tolerance result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:AVERage
value: FetchStruct = driver.multiEval.listPy.segment.seMask.average.
↪fetch(segment = repcap.Segment.Default)
```

Return spectrum emission single value results for segment <no> in list mode. The values described below are returned by FETCh commands. The first four values (reliability to out-of-tolerance result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

6.2.9.7.7.2 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SEMask:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SEMask:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in slots
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %

- Obw: float or bool: float Occupied bandwidth Unit: Hz
- Tx_Power: float or bool: float Total TX power in the slot Unit: dBm

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in slots
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Obw: float: float Occupied bandwidth Unit: Hz
- Tx_Power: float: float Total TX power in the slot Unit: dBm

calculate(*segment=Segment.Default*) → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:CURRent
value: CalculateStruct = driver.multiEval.listPy.segment.seMask.current.
↪calculate(segment = repcap.Segment.Default)
```

Return spectrum emission single value results for segment <no> in list mode. The values described below are returned by FETCh commands. The first four values (reliability to out-of-tolerance result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:CURRent
value: FetchStruct = driver.multiEval.listPy.segment.seMask.current.
↪fetch(segment = repcap.Segment.Default)
```

Return spectrum emission single value results for segment <no> in list mode. The values described below are returned by FETCh commands. The first four values (reliability to out-of-tolerance result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

6.2.9.7.7.3 Dallocation

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SEMask:DALLocation
```

class DallocationCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Nr_Res_Blocks: int: decimal Number of allocated resource blocks
- Offset_Res_Blocks: int: decimal Offset of the first allocated resource block from the edge of the allocated UL transmission bandwidth

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:DALLocation
value: FetchStruct = driver.multiEval.listPy.segment.seMask.dallocation.
↪fetch(segment = repcap.Segment.Default)
```

Return the detected allocation for segment <no> in list mode. The result is determined from the last measured slot of the statistical length. The individual measurements provide the same result when measuring the same slot. However different statistical lengths can be defined for the measurements so that the measured slots and returned results can differ.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

6.2.9.7.7.4 DchType

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SEMask:DCHType
```

class DchTypeCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Channel_Type: enums.UplinkChannelType: PUSCh | PUCCh

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:DCHType
value: FetchStruct = driver.multiEval.listPy.segment.seMask.dchType.
↪fetch(segment = repcap.Segment.Default)
```

Return the uplink channel type for segment <no> in list mode. The result is determined from the last measured slot of the statistical length. The individual measurements provide the same result when measuring the same slot. However different statistical lengths can be defined for the measurements so that the measured slots and returned results can differ.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

6.2.9.7.7.5 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SEMask:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SEMask:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in slots
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Obw: float or bool: float Occupied bandwidth Unit: Hz
- Tx_Power_Min: float or bool: float Minimum total TX power in the slot Unit: dBm
- Tx_Power_Max: float or bool: float Maximum total TX power in the slot Unit: dBm

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in slots
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Obw: float: float Occupied bandwidth Unit: Hz
- Tx_Power_Min: float: float Minimum total TX power in the slot Unit: dBm

- Tx_Power_Max: float: float Maximum total TX power in the slot Unit: dBm

calculate(*segment=Segment.Default*) → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:EXTReme
value: CalculateStruct = driver.multiEval.listPy.segment.seMask.extreme.
↪calculate(segment = repcap.Segment.Default)
```

Return spectrum emission extreme results for segment <no> in list mode. The values described below are returned by FETCh commands. The first four values (reliability to out-of-tolerance result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:EXTReme
value: FetchStruct = driver.multiEval.listPy.segment.seMask.extreme.
↪fetch(segment = repcap.Segment.Default)
```

Return spectrum emission extreme results for segment <no> in list mode. The values described below are returned by FETCh commands. The first four values (reliability to out-of-tolerance result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

6.2.9.7.7.6 Margin

class MarginCls

Margin commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.segment.seMask.margin.clone()
```

Subgroups

6.2.9.7.7.7 All

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SEMask:MARGin:ALL
```

class AllCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in slots
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Margin_Curr_Neg: List[float]: No parameter help available
- Margin_Curr_Pos: List[float]: No parameter help available
- Margin_Avg_Neg: List[float]: No parameter help available
- Margin_Avg_Pos: List[float]: No parameter help available
- Margin_Min_Neg: List[float]: No parameter help available
- Margin_Min_Pos: List[float]: No parameter help available

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:MARGin:ALL
value: FetchStruct = driver.multiEval.listPy.segment.seMask.margin.all.
↪fetch(segment = repcap.Segment.Default)
```

Return limit line margin values, i.e. vertical distances between the spectrum emission mask and a trace, for segment <no> in list mode.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

6.2.9.7.7.8 Average

class AverageCls

Average 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.multiEval.listPy.segment.seMask.margin.average.clone()
```

Subgroups

6.2.9.7.7.9 Negativ

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:MARGin:AVERage:NEGativ
```

class NegativCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in slots
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Margin_Avg_Neg_X: List[float]: No parameter help available
- Margin_Avg_Neg_Y: List[float]: No parameter help available

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:MARGin:AVERage:NEGativ
value: FetchStruct = driver.multiEval.listPy.segment.seMask.margin.average.
↪negativ.fetch(segment = repcap.Segment.Default)
```

Return spectrum emission mask margin results for segment <no> in list mode. The individual commands provide results for the CURRent, AVERage and maximum traces (resulting in MINimum margins) for NEGative and POSitive offset frequencies. Returned sequence: <Reliability>, <SegReliability>, <StatistExpired>, <OutOfTolerance>, {<MarginX>, <MarginY>}area1, {... }area2, ..., {... }area12

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

6.2.9.7.7.10 Positiv

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:MARGin:AVERage:POSitiv
```

class PositivCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in slots
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Margin_Avg_Pos_X: List[float]: No parameter help available
- Margin_Avg_Pos_Y: List[float]: No parameter help available

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:MARGin:AVERage:POSitiv
value: FetchStruct = driver.multiEval.listPy.segment.seMask.margin.average.
↪positiv.fetch(segment = repcap.Segment.Default)
```

Return spectrum emission mask margin results for segment <no> in list mode. The individual commands provide results for the CURRent, AVERage and maximum traces (resulting in MINimum margins) for NEGative and POSitive offset frequencies. Returned sequence: <Reliability>, <SegReliability>, <StatistExpired>, <OutOfTolerance>, {<MarginX>, <MarginY>}area1, {... }area2, ..., {...}area12

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

6.2.9.7.7.11 Current

class CurrentCls

Current 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.multiEval.listPy.segment.seMask.margin.current.clone()
```

Subgroups

6.2.9.7.7.12 Negativ

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:MARGin:CURRent:NEGativ
```

class NegativCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in slots
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Margin_Curr_Neg_X: List[float]: No parameter help available
- Margin_Curr_Neg_Y: List[float]: No parameter help available

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:MARGin:CURRent:NEGativ
value: FetchStruct = driver.multiEval.listPy.segment.seMask.margin.current.
↪negativ.fetch(segment = repcap.Segment.Default)
```

Return spectrum emission mask margin results for segment <no> in list mode. The individual commands provide results for the CURRent, AVERage and maximum traces (resulting in MINimum margins) for NEGative and POSitive offset frequencies. Returned sequence: <Reliability>, <SegReliability>, <StatistExpired>, <OutOfTolerance>, {<MarginX>, <MarginY>}area1, {... }area2, ..., {...}area12

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

6.2.9.7.7.13 Positiv

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:MARGin:CURRent:POSitiv
```

class PositivCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in slots
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Margin_Curr_Pos_X: List[float]: No parameter help available
- Margin_Curr_Pos_Y: List[float]: No parameter help available

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:MARGin:CURRent:POSitiv
value: FetchStruct = driver.multiEval.listPy.segment.seMask.margin.current.
↪positiv.fetch(segment = repcap.Segment.Default)
```

Return spectrum emission mask margin results for segment <no> in list mode. The individual commands provide results for the CURRent, AVERage and maximum traces (resulting in MINimum margins) for NEGative and POSitive offset frequencies. Returned sequence: <Reliability>, <SegReliability>, <StatistExpired>, <OutOfTolerance>, {<MarginX>, <MarginY>}area1, {... }area2, ..., {...}area12

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

6.2.9.7.7.14 Minimum

class MinimumCls

Minimum 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.multiEval.listPy.segment.seMask.margin.minimum.clone()
```

Subgroups

6.2.9.7.7.15 Negativ

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:MARGin:MINimum:NEGativ
```

class NegativCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in slots
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Margin_Min_Neg_X: List[float]: No parameter help available
- Margin_Min_Neg_Y: List[float]: No parameter help available

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:MARGin:MINimum:NEGativ
value: FetchStruct = driver.multiEval.listPy.segment.seMask.margin.minimum.
↪negativ.fetch(segment = repcap.Segment.Default)
```

Return spectrum emission mask margin results for segment <no> in list mode. The individual commands provide results for the CURRent, AVERage and maximum traces (resulting in MINimum margins) for NEGative and POSitive offset frequencies. Returned sequence: <Reliability>, <SegReliability>, <StatistExpired>, <OutOfTolerance>, {<MarginX>, <MarginY>}area1, {... }area2, ..., {...}area12

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

6.2.9.7.7.16 Positiv

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:MARGin:MINimum:POSitiv
```

class PositivCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in slots
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Margin_Min_Pos_X: List[float]: No parameter help available
- Margin_Min_Pos_Y: List[float]: No parameter help available

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:MARGin:MINimum:POSitiv
value: FetchStruct = driver.multiEval.listPy.segment.seMask.margin.minimum.
↪positiv.fetch(segment = repcap.Segment.Default)
```

Return spectrum emission mask margin results for segment <no> in list mode. The individual commands provide results for the CURRent, AVERage and maximum traces (resulting in MINimum margins) for NEGative and POSitive offset frequencies. Returned sequence: <Reliability>, <SegReliability>, <StatistExpired>, <OutOfTolerance>, {<MarginX>, <MarginY>}area1, {... }area2, ..., {... }area12

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

6.2.9.7.7.17 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SEMask:SDEViation
```

class StandardDevCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'

- Seg_Reliability: int: decimal Reliability indicator for the segment
- Statist_Expired: int: decimal Reached statistical length in slots
- Out_Of_Tolerance: int: decimal Percentage of measured subframes with failed limit check Unit: %
- Obw: float: float Occupied bandwidth Unit: Hz
- Tx_Power: float: float Total TX power in the slot Unit: dBm

fetch(*segment*=*Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:SDEViation
value: FetchStruct = driver.multiEval.listPy.segment.seMask.standardDev.
↪fetch(segment = repcap.Segment.Default)
```

Return spectrum emission single value results for segment <no> in list mode. The values described below are returned by FETCh commands. The first four values (reliability to out-of-tolerance result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

param segment
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

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

6.2.9.8 SeMask

class SeMaskCls

SeMask 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.multiEval.listPy.seMask.clone()
```

Subgroups

6.2.9.8.1 Dallocation

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:DALLocation
```

class DallocationCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'

- Nr_Res_Blocks: List[int]: decimal Number of allocated resource blocks
- Offset_Res_Blocks: List[int]: decimal Offset of the first allocated resource block from the edge of the allocated UL transmission bandwidth

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:DALLocation
value: FetchStruct = driver.multiEval.listPy.seMask.dallocation.fetch()
```

Return the detected allocation for all measured list mode segments. The result is determined from the last measured slot of the statistical length of a segment. The individual measurements provide the same result when measuring the same slot. However different statistical lengths can be defined for the measurements so that the measured slots and returned results can differ. The results are returned as pairs per segment: <Reliability>, {<NrResBlocks>, <OffsetResBlocks>}Seg 1, {<NrResBlocks>, <OffsetResBlocks>}Seg 2, ...

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

6.2.9.8.2 DchType

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:DCHType
```

class DchTypeCls

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

fetch() → List[UplinkChannelType]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:DCHType
value: List[enums.UplinkChannelType] = driver.multiEval.listPy.seMask.dchType.
↪fetch()
```

Return the uplink channel type for all measured list mode segments. The result is determined from the last measured slot of the statistical length of a segment. The individual measurements provide the same result when measuring the same slot. However different statistical lengths can be defined for the measurements so that the measured slots and returned results can differ.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
channel_type: PUSCh | PUCCh Comma-separated list of values, one per measured segment

6.2.9.8.3 Margin

class MarginCls

Margin 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.multiEval.listPy.seMask.margin.clone()
```

Subgroups

6.2.9.8.3.1 Area<Area>

RepCap Settings

```
# Range: Nr1 .. Nr12
rc = driver.multiEval.listPy.seMask.margin.area.repcap_area_get()
driver.multiEval.listPy.seMask.margin.area.repcap_area_set(repcap.Area.Nr1)
```

class AreaCls

Area commands group definition. 6 total commands, 2 Subgroups, 0 group commands Repeated Capability: Area, default value after init: Area.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.seMask.margin.area.clone()
```

Subgroups

6.2.9.8.3.2 Negativ

class NegativCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.seMask.margin.area.negativ.clone()
```

Subgroups

6.2.9.8.3.3 Average

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:MARGin:AREA<nr>:NEGativ:AVERage
```

class AverageCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Margin_Avg_Neg_X: List[float]: No parameter help available
- Margin_Avg_Neg_Y: List[float]: No parameter help available

fetch(*area=Area.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:MARGin:AREA<nr>
↪:NEGativ:AVERage
value: FetchStruct = driver.multiEval.listPy.seMask.margin.area.negativ.average.
↪fetch(area = repcap.Area.Default)
```

Return spectrum emission mask margin positions for all measured list mode segments. The individual commands provide results for the CURRent, AVERage and maximum traces (resulting in MINimum margins) for NEGative and POSitive offset frequencies. The results are returned as pairs per segment: <Reliability>, {<MarginPosX>, <MarginPosY>}Seg 1, {<MarginPosX>, <MarginPosY>}Seg 2, ...

param area
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Area')

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

6.2.9.8.3.4 Current

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:MARGin:AREA<nr>:NEGativ:CURRent
```

class CurrentCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Margin_Curr_Neg_X: List[float]: No parameter help available
- Margin_Curr_Neg_Y: List[float]: No parameter help available

fetch(*area=Area.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:MARGin:AREA<nr>
↪:NEGativ:CURRent
value: FetchStruct = driver.multiEval.listPy.seMask.margin.area.negativ.current.
↪fetch(area = repcap.Area.Default)
```

Return spectrum emission mask margin positions for all measured list mode segments. The individual commands provide results for the CURRent, AVERage and maximum traces (resulting in MINimum margins) for NEGative and POSitive offset frequencies. The results are returned as pairs per segment: <Reliability>, {<MarginPosX>, <MarginPosY>}Seg 1, {<MarginPosX>, <MarginPosY>}Seg 2, ...

param area
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Area')

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

6.2.9.8.3.5 Minimum

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:MARGin:AREA<nr>:NEGativ:MINimum
```

class MinimumCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Margin_Min_Neg_X: List[float]: No parameter help available
- Margin_Min_Neg_Y: List[float]: No parameter help available

fetch(*area=Area.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:MARGin:AREA<nr>
↪:NEGativ:MINimum
value: FetchStruct = driver.multiEval.listPy.seMask.margin.area.negativ.minimum.
↪fetch(area = repcap.Area.Default)
```

Return spectrum emission mask margin positions for all measured list mode segments. The individual commands provide results for the CURRent, AVERage and maximum traces (resulting in MINimum margins) for NEGative and POSitive offset frequencies. The results are returned as pairs per segment: <Reliability>, {<MarginPosX>, <MarginPosY>}Seg 1, {<MarginPosX>, <MarginPosY>}Seg 2, ...

param area
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Area')

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

6.2.9.8.3.6 Positiv

class PositivCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.seMask.margin.area.positiv.clone()
```

Subgroups

6.2.9.8.3.7 Average

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:MARGin:AREA<nr>:POSitiv:AVERage
```

class AverageCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Margin_Avg_Pos_X: List[float]: float X position of margin for selected area Unit: Hz
- Margin_Avg_Pos_Y: List[float]: float Y position of margin for selected area Unit: dB

fetch(*area=Area.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:MARGin:AREA<nr>
↪:POSitiv:AVERage
value: FetchStruct = driver.multiEval.listPy.seMask.margin.area.positiv.average.
↪fetch(area = repcap.Area.Default)
```

Return spectrum emission mask margin positions for all measured list mode segments. The individual commands provide results for the CURRent, AVERage and maximum traces (resulting in MINimum margins) for NEGative and POSitive offset frequencies. The results are returned as pairs per segment: <Reliability>, {<MarginPosX>, <MarginPosY>}Seg 1, {<MarginPosX>, <MarginPosY>}Seg 2, ...

param area
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Area')

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

6.2.9.8.3.8 Current

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:MARGin:AREA<nr>:POSitiv:CURRent
```

class CurrentCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Margin_Curr_Pos_X: List[float]: float X position of margin for selected area Unit: Hz
- Margin_Curr_Pos_Y: List[float]: float Y position of margin for selected area Unit: dB

fetch(*area=Area.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:MARGin:AREA<nr>
↪:POSitiv:CURRent
value: FetchStruct = driver.multiEval.listPy.seMask.margin.area.positiv.current.
↪fetch(area = repcap.Area.Default)
```

Return spectrum emission mask margin positions for all measured list mode segments. The individual commands provide results for the CURRent, AVERage and maximum traces (resulting in MINimum margins) for NEGative and POSitive offset frequencies. The results are returned as pairs per segment: <Reliability>, {<MarginPosX>, <MarginPosY>}Seg 1, {<MarginPosX>, <MarginPosY>}Seg 2, ...

param area
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Area')

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

6.2.9.8.3.9 Minimum

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:MARGin:AREA<nr>:POSitiv:MINimum
```

class MinimumCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Margin_Min_Pos_X: List[float]: float X position of margin for selected area Unit: Hz
- Margin_Min_Pos_Y: List[float]: float Y position of margin for selected area Unit: dB

fetch(*area=Area.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:MARGin:AREA<nr>
↪:POSitiv:MINimum
value: FetchStruct = driver.multiEval.listPy.seMask.margin.area.positiv.minimum.
↪fetch(area = repcap.Area.Default)
```

Return spectrum emission mask margin positions for all measured list mode segments. The individual commands provide results for the CURRent, AVERage and maximum traces (resulting in MINimum margins) for NEGative and POSitive offset frequencies. The results are returned as pairs per segment: <Reliability>, {<MarginPosX>, <MarginPosY>}Seg 1, {<MarginPosX>, <MarginPosY>}Seg 2, ...

param area
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Area')

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

6.2.9.8.4 Obw

class ObwCls

Obw commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.9.8.4.1 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:OBW:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:OBW:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:OBW:AVERage
value: List[float or bool] = driver.multiEval.listPy.seMask.obw.average.
↪calculate()
```

Return the occupied bandwidth for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
obw: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: Hz

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:OBW:AVERage
value: List[float] = driver.multiEval.listPy.seMask.obw.average.fetch()
```

Return the occupied bandwidth for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
obw: float Comma-separated list of values, one per measured segment Unit: Hz

6.2.9.8.4.2 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:OBW:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:OBW:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:OBW:CURRent
value: List[float or bool] = driver.multiEval.listPy.seMask.obw.current.
↪calculate()
```

Return the occupied bandwidth for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
obw: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: Hz

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:OBW:CURRent
value: List[float] = driver.multiEval.listPy.seMask.obw.current.fetch()
```

Return the occupied bandwidth for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
obw: float Comma-separated list of values, one per measured segment Unit: Hz

6.2.9.8.4.3 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:OBW:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:OBW:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:OBW:EXTReme
value: List[float or bool] = driver.multiEval.listPy.seMask.obw.extreme.
↪calculate()
```

Return the occupied bandwidth for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
obw: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: Hz

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:OBW:EXTReme
value: List[float] = driver.multiEval.listPy.seMask.obw.extreme.fetch()
```

Return the occupied bandwidth for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
obw: float Comma-separated list of values, one per measured segment Unit: Hz

6.2.9.8.4.4 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:OBW:SDEViation
```

class StandardDevCls

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

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:OBW:SDEViation
value: List[float] = driver.multiEval.listPy.seMask.obw.standardDev.fetch()
```

Return the occupied bandwidth for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
obw: float Comma-separated list of values, one per measured segment Unit: Hz

6.2.9.8.5 TxPower

class TxPowerCls

TxPower commands group definition. 9 total commands, 5 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.9.8.5.1 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:TXPower:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:TXPower:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:SEMask:TXPower:AVERage
value: List[float or bool] = driver.multiEval.listPy.seMask.txPower.average.
↪calculate()
```

Return the total TX power in the slot for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
tx_power: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: dBm

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:TXPower:AVERage
value: List[float] = driver.multiEval.listPy.seMask.txPower.average.fetch()
```

Return the total TX power in the slot for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
tx_power: float Comma-separated list of values, one per measured segment Unit: dBm

6.2.9.8.5.2 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:TXPower:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:TXPower:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:SEMask:TXPower:CURRent
value: List[float or bool] = driver.multiEval.listPy.seMask.txPower.current.
↪calculate()
```

Return the total TX power in the slot for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
tx_power: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: dBm

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:TXPower:CURRent
value: List[float] = driver.multiEval.listPy.seMask.txPower.current.fetch()
```

Return the total TX power in the slot for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
tx_power: float Comma-separated list of values, one per measured segment Unit: dBm

6.2.9.8.5.3 Maximum

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:TXPower:MAXimum
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:TXPower:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:SEMask:TXPower:MAXimum
value: List[float or bool] = driver.multiEval.listPy.seMask.txPower.maximum.
↪calculate()
```

Return the total TX power in the slot for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
tx_power: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: dBm

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:TXPower:MAXimum
value: List[float] = driver.multiEval.listPy.seMask.txPower.maximum.fetch()
```

Return the total TX power in the slot for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
tx_power: float Comma-separated list of values, one per measured segment Unit: dBm

6.2.9.8.5.4 Minimum

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:TXPower:MINimum
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:TXPower:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:SEMask:TXPower:MINimum
value: List[float or bool] = driver.multiEval.listPy.seMask.txPower.minimum.
↪calculate()
```

Return the total TX power in the slot for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
tx_power: (float or boolean items) float Comma-separated list of values, one per measured segment Unit: dBm

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:TXPower:MINimum
value: List[float] = driver.multiEval.listPy.seMask.txPower.minimum.fetch()
```

Return the total TX power in the slot for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
tx_power: float Comma-separated list of values, one per measured segment Unit: dBm

6.2.9.8.5.5 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:TXPower:SDEViation
```

class StandardDevCls

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

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:SEMask:TXPower:SDEViation
value: List[float] = driver.multiEval.listPy.seMask.txPower.standardDev.fetch()
```

Return the total TX power in the slot for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
tx_power: float Comma-separated list of values, one per measured segment Unit: dBm

6.2.9.9 Sreliability

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SRELiability
```

class SreliabilityCls

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

fetch() → List[int]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SRELiability
value: List[int] = driver.multiEval.listPy.sreliability.fetch()
```

Returns the segment reliability for all measured list mode segments. A common reliability indicator of zero 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. If you get a non-zero common reliability indicator, you can use this command to retrieve the individual reliability values of all measured segments for further analysis.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
seg_reliability: decimal Comma-separated list of values, one per measured segment The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.

6.2.10 Merror

class MerrorCls

Merror 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.multiEval.merror.clone()
```

Subgroups

6.2.10.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:MERRor:AVERage
FETCh:LTE:MEASurement<Instance>:MEValuation:MERRor:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Low: List[float]: float Magnitude error value for low EVM window position Unit: %
- High: List[float]: float Magnitude error value for high EVM window position Unit: %

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:MERRor:AVERage
value: ResultData = driver.multiEval.merror.average.fetch()
```

Returns the values of the magnitude error bar graphs for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum bar graphs can be retrieved. There is one pair of magnitude error values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, … See also 'View Magnitude Error, Phase Error'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:MERRor:AVERage
value: ResultData = driver.multiEval.merror.average.read()
```

Returns the values of the magnitude error bar graphs for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum bar graphs can be retrieved. There is one pair of magnitude error values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, … See also 'View Magnitude Error, Phase Error'.

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

6.2.10.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:MERRor:CURRent
FETCh:LTE:MEASurement<Instance>:MEValuation:MERRor:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Low: List[float]: float Magnitude error value for low EVM window position Unit: %
- High: List[float]: float Magnitude error value for high EVM window position Unit: %

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:MERRor:CURRent
value: ResultData = driver.multiEval.merror.current.fetch()
```

Returns the values of the magnitude error bar graphs for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum bar graphs can be retrieved. There is one pair of magnitude error values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'View Magnitude Error, Phase Error'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:MERRor:CURRent
value: ResultData = driver.multiEval.merror.current.read()
```

Returns the values of the magnitude error bar graphs for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum bar graphs can be retrieved. There is one pair of magnitude error values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'View Magnitude Error, Phase Error'.

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

6.2.10.3 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:MERRor:MAXimum
FETCh:LTE:MEASurement<Instance>:MEValuation:MERRor:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Low: List[float]: float Magnitude error value for low EVM window position Unit: %
- High: List[float]: float Magnitude error value for high EVM window position Unit: %

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:MERRor:MAXimum
value: ResultData = driver.multiEval.merror.maximum.fetch()
```

Returns the values of the magnitude error bar graphs for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum bar graphs can be retrieved. There is one pair of magnitude error values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'View Magnitude Error, Phase Error'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:MERRor:MAXimum
value: ResultData = driver.multiEval.merror.maximum.read()
```

Returns the values of the magnitude error bar graphs for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum bar graphs can be retrieved. There is one pair of magnitude error values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'View Magnitude Error, Phase Error'.

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

6.2.11 Modulation

class ModulationCls

Modulation commands group definition. 15 total commands, 8 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.11.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:MODulation:AVERage
FETCh:LTE:MEASurement<Instance>:MEValuation:MODulation:AVERage
CALCulate:LTE: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: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits. Unit: %
- Evm_Rms_Low: float or bool: float EVM RMS value, low EVM window position Unit: %
- Evm_Rms_High: float or bool: float EVM RMS value, high EVM window position Unit: %
- Evm_Peak_Low: float or bool: float EVM peak value, low EVM window position Unit: %
- Evm_Peak_High: float or bool: float EVM peak value, high EVM window position Unit: %
- Mag_Error_Rms_Low: float or bool: float Magnitude error RMS value, low EVM window position Unit: %
- Mag_Error_Rms_High: float or bool: float Magnitude error RMS value, low EVM window position Unit: %
- Mag_Error_Peak_Low: float or bool: float Magnitude error peak value, low EVM window position Unit: %

- Mag_Err_Peak_High: float or bool: float Magnitude error peak value, high EVM window position Unit: %
- Ph_Error_Rms_Low: float or bool: float Phase error RMS value, low EVM window position Unit: deg
- Ph_Error_Rms_High: float or bool: float Phase error RMS value, high EVM window position Unit: deg
- Ph_Error_Peak_Low: float or bool: float Phase error peak value, low EVM window position Unit: deg
- Ph_Error_Peak_High: float or bool: float Phase error peak value, high EVM window position Unit: deg
- Iq_Offset: float or bool: float I/Q origin offset Unit: dBc
- Frequency_Error: float or bool: float Carrier frequency error Unit: Hz
- Timing_Error: float or bool: float Time error Unit: Ts (basic LTE time unit)
- Tx_Power: float or bool: float User equipment power Unit: dBm
- Peak_Power: float or bool: float User equipment peak power Unit: dBm
- Psd: float or bool: No parameter help available
- Evm_Dmrs_Low: float or bool: float EVM DMRS value, low EVM window position Unit: %
- Evm_Dmrs_High: float or bool: float EVM DMRS value, high EVM window position Unit: %
- Mag_Err_Dmrs_Low: float or bool: float Magnitude error DMRS value, low EVM window position Unit: %
- Mag_Err_Dmrs_High: float or bool: float Magnitude error DMRS value, high EVM window position Unit: %
- Ph_Error_Dmrs_Low: float or bool: float Phase error DMRS value, low EVM window position Unit: deg
- Ph_Error_Dmrs_High: float or bool: float Phase error DMRS value, high EVM window position Unit: deg
- Iq_Gain_Imbalance: float or bool: float Gain imbalance Unit: dB
- Iq_Quadrature_Err: float or bool: float Quadrature error Unit: deg
- Evm_Srs: float: float Error vector magnitude result for SRS signals Unit: %

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits. Unit: %
- Evm_Rms_Low: float: float EVM RMS value, low EVM window position Unit: %
- Evm_Rms_High: float: float EVM RMS value, high EVM window position Unit: %
- Evm_Peak_Low: float: float EVM peak value, low EVM window position Unit: %
- Evm_Peak_High: float: float EVM peak value, high EVM window position Unit: %
- Mag_Error_Rms_Low: float: float Magnitude error RMS value, low EVM window position Unit: %
- Mag_Error_Rms_High: float: float Magnitude error RMS value, low EVM window position Unit: %

- Mag_Error_Peak_Low: float: float Magnitude error peak value, low EVM window position Unit: %
- Mag_Err_Peak_High: float: float Magnitude error peak value, high EVM window position Unit: %
- Ph_Error_Rms_Low: float: float Phase error RMS value, low EVM window position Unit: deg
- Ph_Error_Rms_High: float: float Phase error RMS value, high EVM window position Unit: deg
- Ph_Error_Peak_Low: float: float Phase error peak value, low EVM window position Unit: deg
- Ph_Error_Peak_High: float: float Phase error peak value, high EVM window position Unit: deg
- Iq_Offset: float: float I/Q origin offset Unit: dBc
- Frequency_Error: float: float Carrier frequency error Unit: Hz
- Timing_Error: float: float Time error Unit: Ts (basic LTE time unit)
- Tx_Power: float: float User equipment power Unit: dBm
- Peak_Power: float: float User equipment peak power Unit: dBm
- Psd: float: No parameter help available
- Evm_Dmrs_Low: float: float EVM DMRS value, low EVM window position Unit: %
- Evm_Dmrs_High: float: float EVM DMRS value, high EVM window position Unit: %
- Mag_Err_Dmrs_Low: float: float Magnitude error DMRS value, low EVM window position Unit: %
- Mag_Err_Dmrs_High: float: float Magnitude error DMRS value, high EVM window position Unit: %
- Ph_Error_Dmrs_Low: float: float Phase error DMRS value, low EVM window position Unit: deg
- Ph_Error_Dmrs_High: float: float Phase error DMRS value, high EVM window position Unit: deg
- Iq_Gain_Imbalance: float: float Gain imbalance Unit: dB
- Iq_Quadrature_Err: float: float Quadrature error Unit: deg
- Evm_Srs: float: float Error vector magnitude result for SRS signals Unit: %

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:MODulation:AVERage
value: CalculateStruct = driver.multiEval.modulation.average.calculate()
```

Return the current, average and standard deviation single-value 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:LTE:MEASurement<Instance>:MEValuation:MODulation:AVERage
value: ResultData = driver.multiEval.modulation.average.fetch()
```

Return the current, average and standard deviation single-value 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:LTE:MEASurement<Instance>:MEValuation:MODulation:AVERage
value: ResultData = driver.multiEval.modulation.average.read()
```

Return the current, average and standard deviation single-value 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.

6.2.11.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:MODulation:CURRent
FETCh:LTE:MEASurement<Instance>:MEValuation:MODulation:CURRent
CALCulate:LTE: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: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits. Unit: %
- Evm_Rms_Low: float or bool: float EVM RMS value, low EVM window position Unit: %
- Evm_Rms_High: float or bool: float EVM RMS value, high EVM window position Unit: %
- Evm_Peak_Low: float or bool: float EVM peak value, low EVM window position Unit: %
- Evm_Peak_High: float or bool: float EVM peak value, high EVM window position Unit: %
- Mag_Error_Rms_Low: float or bool: float Magnitude error RMS value, low EVM window position Unit: %
- Mag_Error_Rms_High: float or bool: float Magnitude error RMS value, low EVM window position Unit: %
- Mag_Error_Peak_Low: float or bool: float Magnitude error peak value, low EVM window position Unit: %
- Mag_Err_Peak_High: float or bool: float Magnitude error peak value, high EVM window position Unit: %
- Ph_Error_Rms_Low: float or bool: float Phase error RMS value, low EVM window position Unit: deg
- Ph_Error_Rms_High: float or bool: float Phase error RMS value, high EVM window position Unit: deg
- Ph_Error_Peak_Low: float or bool: float Phase error peak value, low EVM window position Unit: deg
- Ph_Error_Peak_High: float or bool: float Phase error peak value, high EVM window position Unit: deg

- Iq_Offset: float or bool: float I/Q origin offset Unit: dBc
- Frequency_Error: float or bool: float Carrier frequency error Unit: Hz
- Timing_Error: float or bool: float Time error Unit: Ts (basic LTE time unit)
- Tx_Power: float or bool: float User equipment power Unit: dBm
- Peak_Power: float or bool: float User equipment peak power Unit: dBm
- Psd: float or bool: No parameter help available
- Evm_Dmrs_Low: float or bool: float EVM DMRS value, low EVM window position Unit: %
- Evm_Dmrs_High: float or bool: float EVM DMRS value, high EVM window position Unit: %
- Mag_Err_Dmrs_Low: float or bool: float Magnitude error DMRS value, low EVM window position Unit: %
- Mag_Err_Dmrs_High: float or bool: float Magnitude error DMRS value, high EVM window position Unit: %
- Ph_Error_Dmrs_Low: float or bool: float Phase error DMRS value, low EVM window position Unit: deg
- Ph_Error_Dmrs_High: float or bool: float Phase error DMRS value, high EVM window position Unit: deg
- Iq_Gain_Imbalance: float or bool: float Gain imbalance Unit: dB
- Iq_Quadrature_Err: float or bool: float Quadrature error Unit: deg
- Evm_Srs: float: float Error vector magnitude result for SRS signals Unit: %

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits. Unit: %
- Evm_Rms_Low: float: float EVM RMS value, low EVM window position Unit: %
- Evm_Rms_High: float: float EVM RMS value, high EVM window position Unit: %
- Evm_Peak_Low: float: float EVM peak value, low EVM window position Unit: %
- Evm_Peak_High: float: float EVM peak value, high EVM window position Unit: %
- Mag_Error_Rms_Low: float: float Magnitude error RMS value, low EVM window position Unit: %
- Mag_Error_Rms_High: float: float Magnitude error RMS value, low EVM window position Unit: %
- Mag_Error_Peak_Low: float: float Magnitude error peak value, low EVM window position Unit: %
- Mag_Err_Peak_High: float: float Magnitude error peak value, high EVM window position Unit: %
- Ph_Error_Rms_Low: float: float Phase error RMS value, low EVM window position Unit: deg
- Ph_Error_Rms_High: float: float Phase error RMS value, high EVM window position Unit: deg
- Ph_Error_Peak_Low: float: float Phase error peak value, low EVM window position Unit: deg
- Ph_Error_Peak_High: float: float Phase error peak value, high EVM window position Unit: deg
- Iq_Offset: float: float I/Q origin offset Unit: dBc
- Frequency_Error: float: float Carrier frequency error Unit: Hz

- Timing_Error: float: float Time error Unit: Ts (basic LTE time unit)
- Tx_Power: float: float User equipment power Unit: dBm
- Peak_Power: float: float User equipment peak power Unit: dBm
- Psd: float: No parameter help available
- Evm_Dmrs_Low: float: float EVM DMRS value, low EVM window position Unit: %
- Evm_Dmrs_High: float: float EVM DMRS value, high EVM window position Unit: %
- Mag_Err_Dmrs_Low: float: float Magnitude error DMRS value, low EVM window position Unit: %
- Mag_Err_Dmrs_High: float: float Magnitude error DMRS value, high EVM window position Unit: %
- Ph_Error_Dmrs_Low: float: float Phase error DMRS value, low EVM window position Unit: deg
- Ph_Error_Dmrs_High: float: float Phase error DMRS value, high EVM window position Unit: deg
- Iq_Gain_Imbalance: float: float Gain imbalance Unit: dB
- Iq_Quadrature_Err: float: float Quadrature error Unit: deg
- Evm_Srs: float: float Error vector magnitude result for SRS signals Unit: %

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:MODulation:CURRent
value: CalculateStruct = driver.multiEval.modulation.current.calculate()
```

Return the current, average and standard deviation single-value 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:LTE:MEASurement<Instance>:MEValuation:MODulation:CURRent
value: ResultData = driver.multiEval.modulation.current.fetch()
```

Return the current, average and standard deviation single-value 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:LTE:MEASurement<Instance>:MEValuation:MODulation:CURRent
value: ResultData = driver.multiEval.modulation.current.read()
```

Return the current, average and standard deviation single-value 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.

6.2.11.3 Dallocation

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:MODulation:DALLocation
```

class DallocationCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Nr_Res_Blocks: int: decimal Number of allocated resource blocks
- Offset_Res_Blocks: int: decimal Offset of the first allocated resource block from the edge of the allocated UL transmission bandwidth

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:MODulation:DALLocation
value: FetchStruct = driver.multiEval.modulation.dallocation.fetch()
```

Returns the detected allocation for the measured slot. If the same slot is measured by the individual measurements, all commands yield the same result. If different statistic counts are defined for the modulation, ACLR and spectrum emission mask measurements, different slots can be measured and different results can be returned by the individual commands.

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

6.2.11.4 DchType

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:MODulation:DCHType
```

class DchTypeCls

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

fetch() → UplinkChannelType

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:MODulation:DCHType
value: enums.UplinkChannelType = driver.multiEval.modulation.dchType.fetch()
```

Returns the uplink channel type for the measured slot. If the same slot is measured by the individual measurements, all commands yield the same result. If different statistic counts are defined for the modulation, ACLR and spectrum emission mask measurements, different slots can be measured and different results can be returned by the individual commands.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
channel_type: PUSCh | PUCCh

6.2.11.5 Dmodulation

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:MODulation:DMODulation
```

class DmodulationCls

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

fetch() → Modulation

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:MODulation:DMODulation
value: enums.Modulation = driver.multiEval.modulation.dmodulation.fetch()
```

Returns the detected modulation scheme in the measured slot. If channel type PUCCH is detected, QPSK is returned as modulation type because the QPSK limits are applied in that case.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
modulation: QPSK | Q16 | Q64 | Q256 QPSK, 16-QAM, 64-QAM, 256-QAM

6.2.11.6 Extreme

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:MODulation:EXTReme
FETCh:LTE:MEASurement<Instance>:MEValuation:MODulation:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:MODulation:EXTReme
```

class ExtremeCls

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

class CalculateStruct

Response structure. Fields:

- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits. Unit: %
- Evm_Rms_Low: float or bool: float EVM RMS value, low EVM window position Unit: %
- Evm_Rms_High: float or bool: float EVM RMS value, high EVM window position Unit: %
- Evm_Peak_Low: float or bool: float EVM peak value, low EVM window position Unit: %
- Evm_Peak_High: float or bool: float EVM peak value, high EVM window position Unit: %
- Mag_Error_Rms_Low: float or bool: float Magnitude error RMS value, low EVM window position Unit: %
- Mag_Error_Rms_High: float or bool: float Magnitude error RMS value, low EVM window position Unit: %
- Mag_Error_Peak_Low: float or bool: float Magnitude error peak value, low EVM window position Unit: %
- Mag_Err_Peak_High: float or bool: float Magnitude error peak value, high EVM window position Unit: %

- Ph_Error_Rms_Low: float or bool: float Phase error RMS value, low EVM window position Unit: deg
- Ph_Error_Rms_High: float or bool: float Phase error RMS value, high EVM window position Unit: deg
- Ph_Error_Peak_Low: float or bool: float Phase error peak value, low EVM window position Unit: deg
- Ph_Error_Peak_High: float or bool: float Phase error peak value, high EVM window position Unit: deg
- Iq_Offset: float or bool: float I/Q origin offset Unit: dBc
- Frequency_Error: float or bool: float Carrier frequency error Unit: Hz
- Timing_Error: float or bool: float Time error Unit: Ts (basic LTE time unit)
- Tx_Power_Minimum: float or bool: float Minimum user equipment power Unit: dBm
- Tx_Power_Maximum: float or bool: float Maximum user equipment power Unit: dBm
- Peak_Power_Min: float or bool: float Minimum user equipment peak power Unit: dBm
- Peak_Power_Max: float or bool: float Maximum user equipment peak power Unit: dBm
- Psd_Minimum: float or bool: No parameter help available
- Psd_Maximum: float or bool: No parameter help available
- Evm_Dmrs_Low: float or bool: float EVM DMRS value, low EVM window position Unit: %
- Evm_Dmrs_High: float or bool: float EVM DMRS value, high EVM window position Unit: %
- Mag_Err_Dmrs_Low: float or bool: float Magnitude error DMRS value, low EVM window position Unit: %
- Mag_Err_Dmrs_High: float or bool: float Magnitude error DMRS value, high EVM window position Unit: %
- Ph_Error_Dmrs_Low: float or bool: float Phase error DMRS value, low EVM window position Unit: deg
- Ph_Error_Dmrs_High: float or bool: float Phase error DMRS value, high EVM window position Unit: deg
- Iq_Gain_Imbalance: float or bool: float Gain imbalance Unit: dB
- Iq_Quadrature_Err: float or bool: float Quadrature error Unit: deg
- Evm_Srs: float: float Error vector magnitude result for SRS signals Unit: %

class ResultData

Response structure. Fields:

- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits. Unit: %
- Evm_Rms_Low: float: float EVM RMS value, low EVM window position Unit: %
- Evm_Rms_High: float: float EVM RMS value, high EVM window position Unit: %
- Evm_Peak_Low: float: float EVM peak value, low EVM window position Unit: %
- Evm_Peak_High: float: float EVM peak value, high EVM window position Unit: %
- Mag_Error_Rms_Low: float: float Magnitude error RMS value, low EVM window position Unit: %
- Mag_Error_Rms_High: float: float Magnitude error RMS value, low EVM window position Unit: %

- Mag_Error_Peak_Low: float: float Magnitude error peak value, low EVM window position Unit: %
- Mag_Err_Peak_High: float: float Magnitude error peak value, high EVM window position Unit: %
- Ph_Error_Rms_Low: float: float Phase error RMS value, low EVM window position Unit: deg
- Ph_Error_Rms_High: float: float Phase error RMS value, high EVM window position Unit: deg
- Ph_Error_Peak_Low: float: float Phase error peak value, low EVM window position Unit: deg
- Ph_Error_Peak_High: float: float Phase error peak value, high EVM window position Unit: deg
- Iq_Offset: float: float I/Q origin offset Unit: dBc
- Frequency_Error: float: float Carrier frequency error Unit: Hz
- Timing_Error: float: float Time error Unit: Ts (basic LTE time unit)
- Tx_Power_Minimum: float: float Minimum user equipment power Unit: dBm
- Tx_Power_Maximum: float: float Maximum user equipment power Unit: dBm
- Peak_Power_Min: float: float Minimum user equipment peak power Unit: dBm
- Peak_Power_Max: float: float Maximum user equipment peak power Unit: dBm
- Psd_Minimum: float: No parameter help available
- Psd_Maximum: float: No parameter help available
- Evm_Dmrs_Low: float: float EVM DMRS value, low EVM window position Unit: %
- Evm_Dmrs_High: float: float EVM DMRS value, high EVM window position Unit: %
- Mag_Err_Dmrs_Low: float: float Magnitude error DMRS value, low EVM window position Unit: %
- Mag_Err_Dmrs_High: float: float Magnitude error DMRS value, high EVM window position Unit: %
- Ph_Error_Dmrs_Low: float: float Phase error DMRS value, low EVM window position Unit: deg
- Ph_Error_Dmrs_High: float: float Phase error DMRS value, high EVM window position Unit: deg
- Iq_Gain_Imbalance: float: float Gain imbalance Unit: dB
- Iq_Quadrature_Err: float: float Quadrature error Unit: deg
- Evm_Srs: float: float Error vector magnitude result for SRS signals Unit: %

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:MODulation:EXTReme
value: CalculateStruct = driver.multiEval.modulation.extreme.calculate()
```

Return the extreme single value 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.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

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

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:MODulation:EXTReme
value: ResultData = driver.multiEval.modulation.extreme.fetch()
```

Return the extreme single value 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.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:MODulation:EXTReme
value: ResultData = driver.multiEval.modulation.extreme.read()
```

Return the extreme single value 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.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

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

6.2.11.7 SchType

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:MODulation:SCHType
```

class SchTypeCls

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

fetch() → SidelinkChannelType

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:MODulation:SCHType
value: enums.SidelinkChannelType = driver.multiEval.modulation.schType.fetch()
```

Returns the sidelink channel type evaluated for modulation results.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
channel_type: PSSCh | PSCCh

6.2.11.8 StandardDev

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:MODulation:SDEViation
FETCh:LTE:MEASurement<Instance>:MEValuation:MODulation:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'

- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits. Unit: %
- Evm_Rms_Low: float: float EVM RMS value, low EVM window position Unit: %
- Evm_Rms_High: float: float EVM RMS value, high EVM window position Unit: %
- Evm_Peak_Low: float: float EVM peak value, low EVM window position Unit: %
- Evm_Peak_High: float: float EVM peak value, high EVM window position Unit: %
- Mag_Error_Rms_Low: float: float Magnitude error RMS value, low EVM window position Unit: %
- Mag_Error_Rms_High: float: float Magnitude error RMS value, low EVM window position Unit: %
- Mag_Error_Peak_Low: float: float Magnitude error peak value, low EVM window position Unit: %
- Mag_Err_Peak_High: float: float Magnitude error peak value, high EVM window position Unit: %
- Ph_Error_Rms_Low: float: float Phase error RMS value, low EVM window position Unit: deg
- Ph_Error_Rms_High: float: float Phase error RMS value, high EVM window position Unit: deg
- Ph_Error_Peak_Low: float: float Phase error peak value, low EVM window position Unit: deg
- Ph_Error_Peak_High: float: float Phase error peak value, high EVM window position Unit: deg
- Iq_Offset: float: float I/Q origin offset Unit: dBc
- Frequency_Error: float: float Carrier frequency error Unit: Hz
- Timing_Error: float: float Time error Unit: Ts (basic LTE time unit)
- Tx_Power: float: float User equipment power Unit: dBm
- Peak_Power: float: float User equipment peak power Unit: dBm
- Psd: float: No parameter help available
- Evm_Dmrs_Low: float: float EVM DMRS value, low EVM window position Unit: %
- Evm_Dmrs_High: float: float EVM DMRS value, high EVM window position Unit: %
- Mag_Err_Dmrs_Low: float: float Magnitude error DMRS value, low EVM window position Unit: %
- Mag_Err_Dmrs_High: float: float Magnitude error DMRS value, high EVM window position Unit: %
- Ph_Error_Dmrs_Low: float: float Phase error DMRS value, low EVM window position Unit: deg
- Ph_Error_Dmrs_High: float: float Phase error DMRS value, high EVM window position Unit: deg
- Iq_Gain_Imbalance: float: float Gain imbalance Unit: dB
- Iq_Quadrature_Err: float: float Quadrature error Unit: deg
- Evm_Srs: float: float Error vector magnitude result for SRS signals Unit: %

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:MODulation:SDEViation
value: ResultData = driver.multiEval.modulation.standardDev.fetch()
```

No command help available

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:MODulation:SDEViation
value: ResultData = driver.multiEval.modulation.standardDev.read()
```

Return the current, average and standard deviation single-value 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.

6.2.12 Pdynamics

class PdynamicsCls

Pdynamics commands group definition. 14 total commands, 5 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.12.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PDYNamics:AVERage
FETCh:LTE:MEASurement<Instance>:MEValuation:PDYNamics:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:PDYNamics:AVERage
```

class AverageCls

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

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for power measurements exceeding the specified power limits. Unit: %
- Off_Power_Before: float or bool: No parameter help available
- On_Power_Rms: float or bool: No parameter help available
- On_Power_Peak: float or bool: No parameter help available
- Off_Power_After: float or bool: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for power measurements exceeding the specified power limits. Unit: %
- Off_Power_Before: float: No parameter help available
- On_Power_Rms: float: No parameter help available
- On_Power_Peak: float: No parameter help available
- Off_Power_After: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:PDYNamics:AVERage
value: CalculateStruct = driver.multiEval.pdynamics.average.calculate()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. A single result table row is returned, from left to right. The meaning of the values depends on the selected time mask, as follows:

Table Header: Time mask / Power1 / Power2 / Power3 / Power4

- General on / off / OFF power (before) / ON power RMS / ON power peak / OFF power (after)
- PUCCH / PUSCH / SRS / SRS ON / ON power RMS / ON power peak / ON power (after)
- SRS blanking / SRS OFF / ON power RMS / ON power peak / ON power (after)

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:LTE:MEASurement<Instance>:MEValuation:PDYNamics:AVERage
value: ResultData = driver.multiEval.pdynamics.average.fetch()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. A single result table row is returned, from left to right. The meaning of the values depends on the selected time mask, as follows:

Table Header: Time mask / Power1 / Power2 / Power3 / Power4

- General on / off / OFF power (before) / ON power RMS / ON power peak / OFF power (after)
- PUCCH / PUSCH / SRS / SRS ON / ON power RMS / ON power peak / ON power (after)
- SRS blanking / SRS OFF / ON power RMS / ON power peak / ON power (after)

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:LTE:MEASurement<Instance>:MEValuation:PDYNamics:AVERage
value: ResultData = driver.multiEval.pdynamics.average.read()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. A single result table row is returned, from left to right. The meaning of the values depends on the selected time mask, as follows:

Table Header: Time mask / Power1 / Power2 / Power3 / Power4

- General on / off / OFF power (before) / ON power RMS / ON power peak / OFF power (after)
- PUCCH / PUSCH / SRS / SRS ON / ON power RMS / ON power peak / ON power (after)
- SRS blanking / SRS OFF / ON power RMS / ON power peak / ON power (after)

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.

6.2.12.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PDYNamics:CURRent
FETCh:LTE:MEASurement<Instance>:MEValuation:PDYNamics:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:PDYNamics:CURRent
```

class CurrentCls

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

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for power measurements exceeding the specified power limits. Unit: %
- Off_Power_Before: float or bool: No parameter help available
- On_Power_Rms: float or bool: No parameter help available
- On_Power_Peak: float or bool: No parameter help available
- Off_Power_After: float or bool: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for power measurements exceeding the specified power limits. Unit: %
- Off_Power_Before: float: No parameter help available
- On_Power_Rms: float: No parameter help available

- On_Power_Peak: float: No parameter help available
- Off_Power_After: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:PDYNamics:CURRent
value: CalculateStruct = driver.multiEval.pdynamics.current.calculate()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. A single result table row is returned, from left to right. The meaning of the values depends on the selected time mask, as follows:

Table Header: Time mask / Power1 / Power2 / Power3 / Power4

- General on / off / OFF power (before) / ON power RMS / ON power peak / OFF power (after)
- PUCCH / PUSCH / SRS / SRS ON / ON power RMS / ON power peak / ON power (after)
- SRS blanking / SRS OFF / ON power RMS / ON power peak / ON power (after)

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:LTE:MEASurement<Instance>:MEValuation:PDYNamics:CURRent
value: ResultData = driver.multiEval.pdynamics.current.fetch()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. A single result table row is returned, from left to right. The meaning of the values depends on the selected time mask, as follows:

Table Header: Time mask / Power1 / Power2 / Power3 / Power4

- General on / off / OFF power (before) / ON power RMS / ON power peak / OFF power (after)
- PUCCH / PUSCH / SRS / SRS ON / ON power RMS / ON power peak / ON power (after)
- SRS blanking / SRS OFF / ON power RMS / ON power peak / ON power (after)

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:LTE:MEASurement<Instance>:MEValuation:PDYNamics:CURRent
value: ResultData = driver.multiEval.pdynamics.current.read()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. A single result table row is returned, from left to right. The meaning of the values depends on the selected time mask, as follows:

Table Header: Time mask / Power1 / Power2 / Power3 / Power4

- General on / off / OFF power (before) / ON power RMS / ON power peak / OFF power (after)

- PUCCH / PUSCH / SRS / SRS ON / ON power RMS / ON power peak / ON power (after)
- SRS blanking / SRS OFF / ON power RMS / ON power peak / ON power (after)

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.

6.2.12.3 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PDYNamics:MAXimum
FETCh:LTE:MEASurement<Instance>:MEValuation:PDYNamics:MAXimum
CALCulate:LTE:MEASurement<Instance>:MEValuation:PDYNamics:MAXimum
```

class MaximumCls

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

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for power measurements exceeding the specified power limits. Unit: %
- Off_Power_Before: float or bool: No parameter help available
- On_Power_Rms: float or bool: No parameter help available
- On_Power_Peak: float or bool: No parameter help available
- Off_Power_After: float or bool: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for power measurements exceeding the specified power limits. Unit: %
- Off_Power_Before: float: No parameter help available
- On_Power_Rms: float: No parameter help available
- On_Power_Peak: float: No parameter help available
- Off_Power_After: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:PDYNamics:MAXimum
value: CalculateStruct = driver.multiEval.pdynamics.maximum.calculate()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. A single result table row is returned, from left to right. The meaning of the values depends on the selected time mask, as follows:

Table Header: Time mask / Power1 / Power2 / Power3 / Power4

- General on / off / OFF power (before) / ON power RMS / ON power peak / OFF power (after)
- PUCCH / PUSCH / SRS / SRS ON / ON power RMS / ON power peak / ON power (after)
- SRS blanking / SRS OFF / ON power RMS / ON power peak / ON power (after)

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:LTE:MEASurement<Instance>:MEValuation:PDYNamics:MAXimum
value: ResultData = driver.multiEval.pdynamics.maximum.fetch()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. A single result table row is returned, from left to right. The meaning of the values depends on the selected time mask, as follows:

Table Header: Time mask / Power1 / Power2 / Power3 / Power4

- General on / off / OFF power (before) / ON power RMS / ON power peak / OFF power (after)
- PUCCH / PUSCH / SRS / SRS ON / ON power RMS / ON power peak / ON power (after)
- SRS blanking / SRS OFF / ON power RMS / ON power peak / ON power (after)

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:LTE:MEASurement<Instance>:MEValuation:PDYNamics:MAXimum
value: ResultData = driver.multiEval.pdynamics.maximum.read()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. A single result table row is returned, from left to right. The meaning of the values depends on the selected time mask, as follows:

Table Header: Time mask / Power1 / Power2 / Power3 / Power4

- General on / off / OFF power (before) / ON power RMS / ON power peak / OFF power (after)
- PUCCH / PUSCH / SRS / SRS ON / ON power RMS / ON power peak / ON power (after)
- SRS blanking / SRS OFF / ON power RMS / ON power peak / ON power (after)

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.

6.2.12.4 Minimum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PDYNamics:MINimum
FETCh:LTE:MEASurement<Instance>:MEValuation:PDYNamics:MINimum
CALCulate:LTE:MEASurement<Instance>:MEValuation:PDYNamics:MINimum
```

class MinimumCls

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

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for power measurements exceeding the specified power limits. Unit: %
- Off_Power_Before: float or bool: No parameter help available
- On_Power_Rms: float or bool: No parameter help available
- On_Power_Peak: float or bool: No parameter help available
- Off_Power_After: float or bool: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for power measurements exceeding the specified power limits. Unit: %
- Off_Power_Before: float: No parameter help available
- On_Power_Rms: float: No parameter help available
- On_Power_Peak: float: No parameter help available
- Off_Power_After: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:PDYNamics:MINimum
value: CalculateStruct = driver.multiEval.pdynamics.minimum.calculate()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. A single result table row is returned, from left to right. The meaning of the values depends on the selected time mask, as follows:

Table Header: Time mask / Power1 / Power2 / Power3 / Power4

- General on / off / OFF power (before) / ON power RMS / ON power peak / OFF power (after)
- PUCCH / PUSCH / SRS / SRS ON / ON power RMS / ON power peak / ON power (after)
- SRS blanking / SRS OFF / ON power RMS / ON power peak / ON power (after)

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:LTE:MEASurement<Instance>:MEValuation:PDYNamics:MINimum
value: ResultData = driver.multiEval.pdynamics.minimum.fetch()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. A single result table row is returned, from left to right. The meaning of the values depends on the selected time mask, as follows:

Table Header: Time mask / Power1 / Power2 / Power3 / Power4

- General on / off / OFF power (before) / ON power RMS / ON power peak / OFF power (after)
- PUCCH / PUSCH / SRS / SRS ON / ON power RMS / ON power peak / ON power (after)
- SRS blanking / SRS OFF / ON power RMS / ON power peak / ON power (after)

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:LTE:MEASurement<Instance>:MEValuation:PDYNamics:MINimum
value: ResultData = driver.multiEval.pdynamics.minimum.read()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. A single result table row is returned, from left to right. The meaning of the values depends on the selected time mask, as follows:

Table Header: Time mask / Power1 / Power2 / Power3 / Power4

- General on / off / OFF power (before) / ON power RMS / ON power peak / OFF power (after)
- PUCCH / PUSCH / SRS / SRS ON / ON power RMS / ON power peak / ON power (after)
- SRS blanking / SRS OFF / ON power RMS / ON power peak / ON power (after)

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.

6.2.12.5 StandardDev

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PDYNamics:SDEViation
FETCh:LTE:MEASurement<Instance>:MEValuation:PDYNamics:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for power measurements exceeding the specified power limits. Unit: %
- Off_Power_Before: float: No parameter help available
- On_Power_Rms: float: No parameter help available
- On_Power_Peak: float: No parameter help available
- Off_Power_After: float: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PDYNamics:SDEViation
value: ResultData = driver.multiEval.pdynamics.standardDev.fetch()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. A single result table row is returned, from left to right. The meaning of the values depends on the selected time mask, as follows:

Table Header: Time mask / Power1 / Power2 / Power3 / Power4

- General on / off / OFF power (before) / ON power RMS / ON power peak / OFF power (after)
- PUCCH / PUSCH / SRS / SRS ON / ON power RMS / ON power peak / ON power (after)
- SRS blanking / SRS OFF / ON power RMS / ON power peak / ON power (after)

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:LTE:MEASurement<Instance>:MEValuation:PDYNamics:SDEViation
value: ResultData = driver.multiEval.pdynamics.standardDev.read()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. A single result table row is returned, from left to right. The meaning of the values depends on the selected time mask, as follows:

Table Header: Time mask / Power1 / Power2 / Power3 / Power4

- General on / off / OFF power (before) / ON power RMS / ON power peak / OFF power (after)
- PUCCH / PUSCH / SRS / SRS ON / ON power RMS / ON power peak / ON power (after)
- SRS blanking / SRS OFF / ON power RMS / ON power peak / ON power (after)

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.

6.2.13 Perror

class PerrorCls

Perror 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.multiEval.perror.clone()
```

Subgroups

6.2.13.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PERRor:AVERage
FETCh:LTE:MEASurement<Instance>:MEValuation:PERRor:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Low: List[float]: float Phase error value for low EVM window position Unit: deg
- High: List[float]: float Phase error value for high EVM window position Unit: deg

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PERRor:AVERage
value: ResultData = driver.multiEval.perror.average.fetch()
```

Returns the values of the phase error bar graphs for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum bar graphs can be retrieved. There is one pair of phase error values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, … See also 'View Magnitude Error, Phase Error'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PERRor:AVERage
value: ResultData = driver.multiEval.perror.average.read()
```

Returns the values of the phase error bar graphs for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum bar graphs can be retrieved. There is one pair of phase error values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, … See also 'View Magnitude Error, Phase Error'.

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

6.2.13.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PERRor:CURRent
FETCh:LTE:MEASurement<Instance>:MEValuation:PERRor:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Low: List[float]: float Phase error value for low EVM window position Unit: deg
- High: List[float]: float Phase error value for high EVM window position Unit: deg

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PERRor:CURRent
value: ResultData = driver.multiEval.perror.current.fetch()
```

Returns the values of the phase error bar graphs for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum bar graphs can be retrieved. There is one pair of phase error values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, … See also 'View Magnitude Error, Phase Error'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PERRor:CURRent
value: ResultData = driver.multiEval.perror.current.read()
```

Returns the values of the phase error bar graphs for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum bar graphs can be retrieved. There is one pair of phase error values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, … See also 'View Magnitude Error, Phase Error'.

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

6.2.13.3 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PERRor:MAXimum
FETCh:LTE:MEASurement<Instance>:MEValuation:PERRor:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Low: List[float]: float Phase error value for low EVM window position Unit: deg
- High: List[float]: float Phase error value for high EVM window position Unit: deg

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PERRor:MAXimum
value: ResultData = driver.multiEval.perror.maximum.fetch()
```

Returns the values of the phase error bar graphs for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum bar graphs can be retrieved. There is one pair of phase error values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, … See also 'View Magnitude Error, Phase Error'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PERRor:MAXimum
value: ResultData = driver.multiEval.perror.maximum.read()
```

Returns the values of the phase error bar graphs for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum bar graphs can be retrieved. There is one pair of phase error values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, … See also 'View Magnitude Error, Phase Error'.

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

6.2.14 Pmonitor

class PmonitorCls

Pmonitor commands group definition. 24 total commands, 6 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.14.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:AVERage
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:PMONitor: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_Tolerance: int: No parameter help available
- Tx_Power: float or bool: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for power measurements exceeding the specified power limits. Unit: %
- Tx_Power: float: float Unit: dBm

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:PMONitor:AVERage
value: CalculateStruct = driver.multiEval.pmonitor.average.calculate()
```

No command help available

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

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:AVERage
value: ResultData = driver.multiEval.pmonitor.average.fetch()
```

Returns the total TX power of all carriers.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:AVERage
value: ResultData = driver.multiEval.pmonitor.average.read()
```

Returns the total TX power of all carriers.

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

6.2.14.2 Cc<CarrierComponent>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.multiEval.pmonitor.cc.repcap_carrierComponent_get()
driver.multiEval.pmonitor.cc.repcap_carrierComponent_set(repcap.CarrierComponent.Nr1)
```

class CcCls

Cc commands group definition. 10 total commands, 5 Subgroups, 0 group commands Repeated Capability: CarrierComponent, default value after init: CarrierComponent.Nr1

Cloning the Group

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

Subgroups

6.2.14.2.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr>:AVERage
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr>:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for power measurements exceeding the specified power limits. Unit: %
- Tx_Power: float: float Unit: dBm

fetch(*carrierComponent=CarrierComponent.Default*) → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr>:AVERage
value: ResultData = driver.multiEval.pmonitor.cc.average.fetch(carrierComponent␣
↪= repcap.CarrierComponent.Default)
```

Returns the TX power of carrier CC<no>.

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

read(*carrierComponent=CarrierComponent.Default*) → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr>:AVERage
value: ResultData = driver.multiEval.pmonitor.cc.average.read(carrierComponent␣
↪= repcap.CarrierComponent.Default)
```

Returns the TX power of carrier CC<no>.

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

6.2.14.2.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr>:CURRent
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr>:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for power measurements exceeding the specified power limits. Unit: %
- Tx_Power: float: float Unit: dBm

fetch(*carrierComponent=CarrierComponent.Default*) → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr>:CURRent
value: ResultData = driver.multiEval.pmonitor.cc.current.fetch(carrierComponent␣
↪= repcap.CarrierComponent.Default)
```

Returns the TX power of carrier CC<no>.

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

read(*carrierComponent=CarrierComponent.Default*) → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr>:CURRent
value: ResultData = driver.multiEval.pmonitor.cc.current.read(carrierComponent␣
↪= repcap.CarrierComponent.Default)
```

Returns the TX power of carrier CC<no>.

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

6.2.14.2.3 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr>:MAXimum
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr>:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for power measurements exceeding the specified power limits. Unit: %
- Tx_Power: float: float Unit: dBm

fetch(*carrierComponent=CarrierComponent.Default*) → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr>:MAXimum
value: ResultData = driver.multiEval.pmonitor.cc.maximum.fetch(carrierComponent␣
↪= repcap.CarrierComponent.Default)
```

Returns the TX power of carrier CC<no>.

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

read(*carrierComponent=CarrierComponent.Default*) → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr>:MAXimum
value: ResultData = driver.multiEval.pmonitor.cc.maximum.read(carrierComponent␣
↪= repcap.CarrierComponent.Default)
```

Returns the TX power of carrier CC<no>.

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

6.2.14.2.4 Minimum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr>:MINimum
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr>:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for power measurements exceeding the specified power limits. Unit: %
- Tx_Power: float: float Unit: dBm

fetch(*carrierComponent=CarrierComponent.Default*) → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr>:MINimum
value: ResultData = driver.multiEval.pmonitor.cc.minimum.fetch(carrierComponent␣
↪= repcap.CarrierComponent.Default)
```

Returns the TX power of carrier CC<no>.

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

read(*carrierComponent=CarrierComponent.Default*) → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr>:MINimum
value: ResultData = driver.multiEval.pmonitor.cc.minimum.read(carrierComponent␣
↪= repcap.CarrierComponent.Default)
```

Returns the TX power of carrier CC<no>.

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

6.2.14.2.5 StandardDev

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr>:SDEViation
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr>:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for power measurements exceeding the specified power limits. Unit: %
- Tx_Power: float: float Unit: dBm

fetch(*carrierComponent=CarrierComponent.Default*) → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr>:SDEViation
value: ResultData = driver.multiEval.pmonitor.cc.standardDev.
↪fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Returns the TX power of carrier CC<no>.

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

read(*carrierComponent=CarrierComponent.Default*) → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr>:SDEViation
value: ResultData = driver.multiEval.pmonitor.cc.standardDev.
↪read(carrierComponent = repcap.CarrierComponent.Default)
```

Returns the TX power of carrier CC<no>.

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

6.2.14.3 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:CURRent
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:PMONitor: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_Tolerance: int: No parameter help available
- Tx_Power: float or bool: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for power measurements exceeding the specified power limits. Unit: %
- Tx_Power: float: float Unit: dBm

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:PMONitor:CURRent
value: CalculateStruct = driver.multiEval.pmonitor.current.calculate()
```

No command help available

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

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:CURRent
value: ResultData = driver.multiEval.pmonitor.current.fetch()
```

Returns the total TX power of all carriers.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:CURRent
value: ResultData = driver.multiEval.pmonitor.current.read()
```

Returns the total TX power of all carriers.

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

6.2.14.4 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:MAXimum
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:MAXimum
CALCulate:LTE:MEASurement<Instance>:MEValuation:PMONitor: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_Tolerance: int: No parameter help available
- Tx_Power: float or bool: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for power measurements exceeding the specified power limits. Unit: %
- Tx_Power: float: float Unit: dBm

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:PMONitor:MAXimum
value: CalculateStruct = driver.multiEval.pmonitor.maximum.calculate()
```

No command help available

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

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:MAXimum
value: ResultData = driver.multiEval.pmonitor.maximum.fetch()
```

Returns the total TX power of all carriers.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:MAXimum
value: ResultData = driver.multiEval.pmonitor.maximum.read()
```

Returns the total TX power of all carriers.

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

6.2.14.5 Minimum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:MINimum
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:MINimum
CALCulate:LTE:MEASurement<Instance>:MEValuation:PMONitor: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_Tolerance: int: No parameter help available
- Tx_Power: float or bool: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for power measurements exceeding the specified power limits. Unit: %
- Tx_Power: float: float Unit: dBm

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:PMONitor:MINimum
value: CalculateStruct = driver.multiEval.pmonitor.minimum.calculate()
```

No command help available

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

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:MINimum
value: ResultData = driver.multiEval.pmonitor.minimum.fetch()
```

Returns the total TX power of all carriers.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:MINimum
value: ResultData = driver.multiEval.pmonitor.minimum.read()
```

Returns the total TX power of all carriers.

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

6.2.14.6 StandardDev

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:SDEViation
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for power measurements exceeding the specified power limits. Unit: %
- Tx_Power: float: float Unit: dBm

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:SDEViation
value: ResultData = driver.multiEval.pmonitor.standardDev.fetch()
```

Returns the total TX power of all carriers.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:SDEViation
value: ResultData = driver.multiEval.pmonitor.standardDev.read()
```

Returns the total TX power of all carriers.

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

6.2.15 ReferenceMarker

class ReferenceMarkerCls

ReferenceMarker commands group definition. 6 total commands, 5 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.15.1 EvMagnitude

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:REFMarker:EVMagnitude
```

class EvMagnitudeCls

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

fetch(*xvalue: int*, *trace_select: TraceSelect*) → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:REFMarker:EVMagnitude
value: float = driver.multiEval.referenceMarker.evMagnitude.fetch(xvalue = 1,␣
↪trace_select = enums.TraceSelect.AVERage)
```

Uses the reference marker on the bar graphs: EVM RMS, EVM peak, magnitude error and phase error vs SC-FDMA symbol.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param xvalue
(integer or boolean) integer Absolute X value of the marker position There are two X values per SC-FDMA symbol on the X axis (symbol 0 low, symbol 0 high, ..., symbol 6 low, symbol 6 high) . Range: 0 to 13

param trace_select
CURRent | AVERage | MAXimum

return
yvalue: float Absolute Y value of the marker position

Cloning the Group

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

Subgroups

6.2.15.1.1 Peak

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:REFMarker:EVMagnitude:PEAK
```

class PeakCls

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

fetch(*xvalue: int*, *trace_select: TraceSelect*) → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:REFMarker:EVMagnitude:PEAK
value: float = driver.multiEval.referenceMarker.evMagnitude.peak.fetch(xvalue =␣
↪1, trace_select = enums.TraceSelect.AVERage)
```

Uses the reference marker on the bar graphs: EVM RMS, EVM peak, magnitude error and phase error vs SC-FDMA symbol.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param xvalue
(integer or boolean) integer Absolute X value of the marker position There are two X values per SC-FDMA symbol on the X axis (symbol 0 low, symbol 0 high, ..., symbol 6 low, symbol 6 high) . Range: 0 to 13

param trace_select
CURRent | AVERage | MAXimum

return
yvalue: float Absolute Y value of the marker position

6.2.15.2 Merror

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:REFMarker:MERRor
```

class MerrorCls

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

fetch(*xvalue: int*, *trace_select: TraceSelect*) → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:REFMarker:MERRor
value: float = driver.multiEval.referenceMarker.merror.fetch(xvalue = 1, trace_
↪select = enums.TraceSelect.AVERage)
```

Uses the reference marker on the bar graphs: EVM RMS, EVM peak, magnitude error and phase error vs SC-FDMA symbol.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param xvalue
(integer or boolean) integer Absolute X value of the marker position There are two X values per SC-FDMA symbol on the X axis (symbol 0 low, symbol 0 high, ..., symbol 6 low, symbol 6 high) . Range: 0 to 13

param trace_select
CURRent | AVERage | MAXimum

return
yvalue: float Absolute Y value of the marker position

6.2.15.3 Pdynamics

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:REFMarker:PDYNamics
```

class PdynamicsCls

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

fetch(*xvalue: float*, *trace_select: TraceSelect*) → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:REFMarker:PDYNamics
value: float = driver.multiEval.referenceMarker.pdynamics.fetch(xvalue = 1.0,␣
↪trace_select = enums.TraceSelect.AVERage)
```

Uses the reference marker on the power dynamics trace.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param xvalue
(float or boolean) integer Absolute X value of the marker position Range: -1100 µs to 2500 µs, Unit: µs

param trace_select
CURRent | AVERage | MAXimum

return
yvalue: float Absolute Y value of the marker position Unit: dBm

6.2.15.4 Perror

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:REFMarker:PERRor
```

class PerrorCls

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

fetch(*xvalue: int*, *trace_select: TraceSelect*) → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:REFMarker:PERRor
value: float = driver.multiEval.referenceMarker.perror.fetch(xvalue = 1, trace_
↪select = enums.TraceSelect.AVERage)
```

Uses the reference marker on the bar graphs: EVM RMS, EVM peak, magnitude error and phase error vs SC-FDMA symbol.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param xvalue
(integer or boolean) integer Absolute X value of the marker position There are two X values per SC-FDMA symbol on the X axis (symbol 0 low, symbol 0 high, ..., symbol 6 low, symbol 6 high) . Range: 0 to 13

param trace_select
CURRent | AVERage | MAXimum

return
yvalue: float Absolute Y value of the marker position

6.2.15.5 Pmonitor

class PmonitorCls

Pmonitor 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.multiEval.referenceMarker.pmonitor.clone()
```

Subgroups

6.2.15.5.1 Cc<CarrierComponent>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.multiEval.referenceMarker.pmonitor.cc.repcap_carrierComponent_get()
driver.multiEval.referenceMarker.pmonitor.cc.repcap_carrierComponent_set(repcap.
↪CarrierComponent.Nr1)
```

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:REFMarker:PMONitor:CC<Nr>
```

class CcCls

Cc commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: CarrierComponent, default value after init: CarrierComponent.Nr1

fetch(*xvalue: int*, *carrierComponent=CarrierComponent.Default*) → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:REFMarker:PMONitor:CC<Nr>
value: float = driver.multiEval.referenceMarker.pmonitor.cc.fetch(xvalue = 1,␣
↪carrierComponent = repcap.CarrierComponent.Default)
```

Uses the reference marker on the power monitor trace.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param xvalue
(integer or boolean) integer Absolute X value of the marker position (subframe number) Range: 0 to 319

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
yvalue: float Absolute Y value of the marker position Unit: dBm

Cloning the Group

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

6.2.16 SeMask

class SeMaskCls

SeMask commands group definition. 20 total commands, 7 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.16.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:SEMask:AVERage
FETCh:LTE:MEASurement<Instance>:MEValuation:SEMask:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:SEMask:AVERage
```

class AverageCls

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

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified spectrum emission mask limits. Unit: %
- Obw: float or bool: float Occupied bandwidth Unit: Hz
- Tx_Power: float or bool: float Total TX power in the slot over all component carriers Unit: dBm

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified spectrum emission mask limits. Unit: %
- Obw: float: float Occupied bandwidth Unit: Hz
- Tx_Power: float: float Total TX power in the slot over all component carriers Unit: dBm

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:SEMask:AVERage
value: CalculateStruct = driver.multiEval.seMask.average.calculate()
```

Return the current, average and standard deviation single-value results of the spectrum emission measurement. 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:LTE:MEASurement<Instance>:MEValuation:SEMask:AVERage
value: ResultData = driver.multiEval.seMask.average.fetch()
```

Return the current, average and standard deviation single-value results of the spectrum emission measurement. 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:LTE:MEASurement<Instance>:MEValuation:SEMask:AVERage
value: ResultData = driver.multiEval.seMask.average.read()
```

Return the current, average and standard deviation single-value results of the spectrum emission measurement. 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.

6.2.16.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:SEMask:CURRent
FETCh:LTE:MEASurement<Instance>:MEValuation:SEMask:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:SEMask:CURRent
```

class CurrentCls

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

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified spectrum emission mask limits. Unit: %
- Obw: float or bool: float Occupied bandwidth Unit: Hz

- Tx_Power: float or bool: float Total TX power in the slot over all component carriers Unit: dBm

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified spectrum emission mask limits. Unit: %
- Obw: float: float Occupied bandwidth Unit: Hz
- Tx_Power: float: float Total TX power in the slot over all component carriers Unit: dBm

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:SEMask:CURRent
value: CalculateStruct = driver.multiEval.seMask.current.calculate()
```

Return the current, average and standard deviation single-value results of the spectrum emission measurement. 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:LTE:MEASurement<Instance>:MEValuation:SEMask:CURRent
value: ResultData = driver.multiEval.seMask.current.fetch()
```

Return the current, average and standard deviation single-value results of the spectrum emission measurement. 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:LTE:MEASurement<Instance>:MEValuation:SEMask:CURRent
value: ResultData = driver.multiEval.seMask.current.read()
```

Return the current, average and standard deviation single-value results of the spectrum emission measurement. 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.

6.2.16.3 Dallocation

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:SEMask:DALLocation
```

class DallocationCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Nr_Res_Blocks: int: decimal Number of allocated resource blocks
- Offset_Res_Blocks: int: decimal Offset of the first allocated resource block from the edge of the allocated UL transmission bandwidth

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:SEMask:DALLocation
value: FetchStruct = driver.multiEval.seMask.dallocation.fetch()
```

Returns the detected allocation for the measured slot. If the same slot is measured by the individual measurements, all commands yield the same result. If different statistic counts are defined for the modulation, ACLR and spectrum emission mask measurements, different slots can be measured and different results can be returned by the individual commands.

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

6.2.16.4 DchType

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:SEMask:DCHType
```

class DchTypeCls

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

fetch() → UplinkChannelType

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:SEMask:DCHType
value: enums.UplinkChannelType = driver.multiEval.seMask.dchType.fetch()
```

Returns the uplink channel type for the measured slot. If the same slot is measured by the individual measurements, all commands yield the same result. If different statistic counts are defined for the modulation, ACLR and spectrum emission mask measurements, different slots can be measured and different results can be returned by the individual commands.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
channel_type: PUSCh | PUCCh

6.2.16.5 Extreme

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:SEMask:EXTReme
FETCh:LTE:MEASurement<Instance>:MEValuation:SEMask:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:SEMask:EXTReme
```

class ExtremeCls

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

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified spectrum emission mask limits. Unit: %
- Obw: float or bool: float Occupied bandwidth Unit: Hz
- Tx_Power_Min: float or bool: float Minimum total TX power in the slot Unit: dBm
- Tx_Power_Max: float or bool: float Maximum total TX power in the slot Unit: dBm

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified spectrum emission mask limits. Unit: %
- Obw: float: float Occupied bandwidth Unit: Hz
- Tx_Power_Min: float: float Minimum total TX power in the slot Unit: dBm
- Tx_Power_Max: float: float Maximum total TX power in the slot Unit: dBm

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:SEMask:EXTReme
value: CalculateStruct = driver.multiEval.seMask.extreme.calculate()
```

Return the extreme single-value results of the spectrum emission measurement. 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:LTE:MEASurement<Instance>:MEValuation:SEMask:EXTReme
value: ResultData = driver.multiEval.seMask.extreme.fetch()
```

Return the extreme single-value results of the spectrum emission measurement. 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:LTE:MEASurement<Instance>:MEValuation:SEMask:EXTReme
value: ResultData = driver.multiEval.seMask.extreme.read()
```

Return the extreme single-value results of the spectrum emission measurement. 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.

6.2.16.6 Margin

class MarginCls

Margin commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.16.6.1 All

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:SEMask:MARGin:ALL
```

class AllCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified spectrum emission mask limits. Unit: %
- Margin_Curr_Neg: List[float]: No parameter help available
- Margin_Curr_Pos: List[float]: No parameter help available
- Margin_Avg_Neg: List[float]: No parameter help available
- Margin_Avg_Pos: List[float]: No parameter help available
- Margin_Min_Neg: List[float]: No parameter help available

- Margin_Min_Pos: List[float]: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:SEMask:MARGin:ALL
value: FetchStruct = driver.multiEval.seMask.margin.all.fetch()
```

Returns spectrum emission mask margin results. A negative margin indicates that the trace is located above the limit line, i.e. the limit is exceeded. Results are provided for the current, average and maximum traces. For each trace, 24 values related to the negative (Neg) and positive (Pos) offset frequencies of emission mask areas 1 to 12 are provided. For inactive areas, NCAP is returned.

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

6.2.16.6.2 Average

class AverageCls

Average 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.multiEval.seMask.margin.average.clone()
```

Subgroups

6.2.16.6.2.1 Negativ

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:SEMask:MARGin:AVERage:NEGativ
```

class NegativCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified spectrum emission mask limits. Unit: %
- Margin_Avg_Neg_X: List[float]: No parameter help available
- Margin_Avg_Neg_Y: List[float]: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:SEMask:MARGin:AVERage:NEGativ
value: FetchStruct = driver.multiEval.seMask.margin.average.negativ.fetch()
```

Returns spectrum emission mask margin results. A negative margin indicates that the trace is located above the limit line, i.e. the limit is exceeded. The individual commands provide results for the CURRent, AVERage and maximum traces (resulting in MINimum margins) . For each trace, the X and Y values of the margins for emission mask areas 1 to 12 are provided for NEGative and POSitive offset frequencies. For inactive areas, NCAP is returned. Returned sequence: <Reliability>, <OutOfTolerance>, {<MarginX>, <MarginY>}area1, {...}area2, ..., {...}area12

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

6.2.16.6.2.2 Positiv

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:SEMask:MARGin:AVERage:POSitiv
```

class PositivCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified spectrum emission mask limits. Unit: %
- Margin_Avg_Pos_X: List[float]: No parameter help available
- Margin_Avg_Pos_Y: List[float]: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:SEMask:MARGin:AVERage:POSitiv
value: FetchStruct = driver.multiEval.seMask.margin.average.positiv.fetch()
```

Returns spectrum emission mask margin results. A negative margin indicates that the trace is located above the limit line, i.e. the limit is exceeded. The individual commands provide results for the CURRent, AVERage and maximum traces (resulting in MINimum margins) . For each trace, the X and Y values of the margins for emission mask areas 1 to 12 are provided for NEGative and POSitive offset frequencies. For inactive areas, NCAP is returned. Returned sequence: <Reliability>, <OutOfTolerance>, {<MarginX>, <MarginY>}area1, {...}area2, ..., {...}area12

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

6.2.16.6.3 Current

class CurrentCls

Current 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.multiEval.seMask.margin.current.clone()
```

Subgroups

6.2.16.6.3.1 Negativ

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:SEMask:MARGin:CURRent:NEGativ
```

class NegativCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified spectrum emission mask limits. Unit: %
- Margin_Curr_Neg_X: List[float]: No parameter help available
- Margin_Curr_Neg_Y: List[float]: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:SEMask:MARGin:CURRent:NEGativ
value: FetchStruct = driver.multiEval.seMask.margin.current.negativ.fetch()
```

Returns spectrum emission mask margin results. A negative margin indicates that the trace is located above the limit line, i.e. the limit is exceeded. The individual commands provide results for the CURRent, AVERage and maximum traces (resulting in MINimum margins) . For each trace, the X and Y values of the margins for emission mask areas 1 to 12 are provided for NEGative and POSitive offset frequencies. For inactive areas, NCAP is returned. Returned sequence: <Reliability>, <OutOfTolerance>, {<MarginX>, <MarginY>}area1, {...}area2, ..., {...}area12

return

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

6.2.16.6.3.2 Positiv

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:SEMask:MARGin:CURRent:POSitiv
```

class PositivCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified spectrum emission mask limits. Unit: %
- Margin_Curr_Pos_X: List[float]: No parameter help available
- Margin_Curr_Pos_Y: List[float]: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:SEMask:MARGin:CURRent:POSitiv
value: FetchStruct = driver.multiEval.seMask.margin.current.positiv.fetch()
```

Returns spectrum emission mask margin results. A negative margin indicates that the trace is located above the limit line, i.e. the limit is exceeded. The individual commands provide results for the CURRent, AVERage and maximum traces (resulting in MINimum margins) . For each trace, the X and Y values of the margins for emission mask areas 1 to 12 are provided for NEGative and POSitive offset frequencies. For inactive areas, NCAP is returned. Returned sequence: <Reliability>, <OutOfTolerance>, {<MarginX>, <MarginY>}area1, {...}area2, ..., {...}area12

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

6.2.16.6.4 Minimum

class MinimumCls

Minimum 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.multiEval.seMask.margin.minimum.clone()
```

Subgroups

6.2.16.6.4.1 Negativ

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:SEMask:MARGin:MINimum:NEGativ
```

class NegativCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified spectrum emission mask limits. Unit: %
- Margin_Min_Neg_X: List[float]: No parameter help available
- Margin_Min_Neg_Y: List[float]: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:SEMask:MARGin:MINimum:NEGativ
value: FetchStruct = driver.multiEval.seMask.margin.minimum.negativ.fetch()
```

Returns spectrum emission mask margin results. A negative margin indicates that the trace is located above the limit line, i.e. the limit is exceeded. The individual commands provide results for the CURRent, AVERage and maximum traces (resulting in MINimum margins) . For each trace, the X and Y values of the margins for emission mask areas 1 to 12 are provided for NEGative and POSitive offset frequencies. For inactive areas, NCAP is returned. Returned sequence: <Reliability>, <OutOfTolerance>, {<MarginX>, <MarginY>}area1, {...}area2, ..., {...}area12

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

6.2.16.6.4.2 Positiv

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:SEMask:MARGin:MINimum:POSitiv
```

class PositivCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'

- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified spectrum emission mask limits. Unit: %
- Margin_Min_Pos_X: List[float]: No parameter help available
- Margin_Min_Pos_Y: List[float]: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:SEMask:MARGin:MINimum:POSitiv
value: FetchStruct = driver.multiEval.seMask.margin.minimum.positiv.fetch()
```

Returns spectrum emission mask margin results. A negative margin indicates that the trace is located above the limit line, i.e. the limit is exceeded. The individual commands provide results for the CURRent, AVERage and maximum traces (resulting in MINimum margins) . For each trace, the X and Y values of the margins for emission mask areas 1 to 12 are provided for NEGative and POSitive offset frequencies. For inactive areas, NCAP is returned. Returned sequence: <Reliability>, <OutOfTolerance>, {<MarginX>, <MarginY>}area1, {...}area2, ..., {...}area12

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

6.2.16.7 StandardDev

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:SEMask:SDEViation
FETCh:LTE:MEASurement<Instance>:MEValuation:SEMask:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified spectrum emission mask limits. Unit: %
- Obw: float: float Occupied bandwidth Unit: Hz
- Tx_Power: float: float Total TX power in the slot over all component carriers Unit: dBm

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:SEMask:SDEViation
value: ResultData = driver.multiEval.seMask.standardDev.fetch()
```

Return the current, average and standard deviation single-value results of the spectrum emission measurement. 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:LTE:MEASurement<Instance>:MEValuation:SEMask:SDEViation
value: ResultData = driver.multiEval.seMask.standardDev.read()
```

Return the current, average and standard deviation single-value results of the spectrum emission measurement. 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.

6.2.17 State

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:STATe
```

class StateCls

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

fetch(*timeout: float = None*, *target_main_state: TargetMainState = None*, *target_sync_state: TargetSyncState = None*) → ResourceState

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:STATe
value: enums.ResourceState = driver.multiEval.state.fetch(timeout = 1.0, target_
↪main_state = enums.TargetMainState.OFF, target_sync_state = enums.
↪TargetSyncState.ADJusted)
```

Queries the main measurement state. Use FETCh:...:STATe:ALL? to query the measurement state including the substates. Use INITiate..., STOP..., ABORt... to change the measurement state.

param timeout
No help available

param target_main_state
No help available

param target_sync_state
No help available

return
meas_status: OFF | RUN | RDY OFF: measurement off, no resources allocated, no results RUN: measurement running, synchronization pending or adjusted, resources active or queued RDY: measurement terminated, valid results can be available

Cloning the Group

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

Subgroups

6.2.17.1 All

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:STATe:ALL
```

class AllCls

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

fetch(*timeout: float = None*, *target_main_state: TargetMainState = None*, *target_sync_state: TargetSyncState = None*) → List[ResourceState]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:STATe:ALL
value: List[enums.ResourceState] = driver.multiEval.state.all.fetch(timeout = 1.
↪0, target_main_state = enums.TargetMainState.OFF, target_sync_state = enums.
↪TargetSyncState.ADJusted)
```

Queries the main measurement state and the measurement substates. Both measurement substates are relevant for running measurements only. Use FETCh:...:STATe? to query the main measurement state only. Use INITiate..., STOP..., ABORt... to change the measurement state.

param timeout
No help available

param target_main_state
No help available

param target_sync_state
No help available

return
state: No help available

6.2.18 Trace

class TraceCls

Trace commands group definition. 36 total commands, 10 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.18.1 Aclr

class AclrCls

Aclr 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.multiEval.trace.aclr.clone()
```

Subgroups

6.2.18.1.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:ACLR:AVERage
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:ACLR:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Utra_2_Neg: float: float Power in the second UTRA channel with lower frequency Unit: dBm
- Utra_1_Neg: float: float Power in the first UTRA channel with lower frequency Unit: dBm
- Eutra_Negativ: float: float Power in the first E-UTRA channel with lower frequency Unit: dBm
- Eutra: float: float Power in the allocated E-UTRA channel Unit: dBm
- Eutra_Positiv: float: float Power in the first E-UTRA channel with higher frequency Unit: dBm
- Utra_1_Pos: float: float Power in the first UTRA channel with higher frequency Unit: dBm
- Utra_2_Pos: float: float Power in the second UTRA channel with higher frequency Unit: dBm

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:ACLR:AVERage
value: ResultData = driver.multiEval.trace.aclr.average.fetch()
```

Returns the absolute powers as displayed in the ACLR diagram. The current and average values can be retrieved. See also 'View Spectrum ACLR'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:ACLR:AVERage
value: ResultData = driver.multiEval.trace.aclr.average.read()
```

Returns the absolute powers as displayed in the ACLR diagram. The current and average values can be retrieved. See also 'View Spectrum ACLR'.

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

6.2.18.1.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:ACLR:CURRent
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:ACLR:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Utra_2_Neg: float: float Power in the second UTRA channel with lower frequency Unit: dBm
- Utra_1_Neg: float: float Power in the first UTRA channel with lower frequency Unit: dBm
- Eutra_Negativ: float: float Power in the first E-UTRA channel with lower frequency Unit: dBm
- Eutra: float: float Power in the allocated E-UTRA channel Unit: dBm
- Eutra_Positiv: float: float Power in the first E-UTRA channel with higher frequency Unit: dBm
- Utra_1_Pos: float: float Power in the first UTRA channel with higher frequency Unit: dBm
- Utra_2_Pos: float: float Power in the second UTRA channel with higher frequency Unit: dBm

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:ACLR:CURRent
value: ResultData = driver.multiEval.trace.aclr.current.fetch()
```

Returns the absolute powers as displayed in the ACLR diagram. The current and average values can be retrieved. See also 'View Spectrum ACLR'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:ACLR:CURRent
value: ResultData = driver.multiEval.trace.aclr.current.read()
```

Returns the absolute powers as displayed in the ACLR diagram. The current and average values can be retrieved. See also 'View Spectrum ACLR'.

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

6.2.18.2 EsFlatness

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:ESFLatness
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:ESFLatness
```

class EsFlatnessCls

EsFlatness commands group definition. 4 total commands, 1 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:ESFLatness
value: List[float] = driver.multiEval.trace.esFlatness.fetch()
```

Returns the values of the equalizer spectrum flatness trace. See also 'View Equalizer Spectrum Flatness'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
power: float Comma-separated list of power values, one value per subcarrier For not allocated subcarriers, NCAP is returned. Unit: dB

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:ESFLatness
value: List[float] = driver.multiEval.trace.esFlatness.read()
```

Returns the values of the equalizer spectrum flatness trace. See also 'View Equalizer Spectrum Flatness'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
power: float Comma-separated list of power values, one value per subcarrier For not allocated subcarriers, NCAP is returned. Unit: dB

Cloning the Group

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

Subgroups

6.2.18.2.1 Phase

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:ESFLatness:PHASe
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:ESFLatness:PHASe
```

class PhaseCls

Phase commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:ESFLatness:PHASe
value: List[float] = driver.multiEval.trace.esFlatness.phase.fetch()
```

Returns the phase values of the equalizer spectrum flatness trace. The GUI shows only the magnitude values.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
phase: float Comma-separated list of phase values, one value per subcarrier For not allocated subcarriers, NCAP is returned. Unit: deg

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:ESFLatness:PHASe
value: List[float] = driver.multiEval.trace.esFlatness.phase.read()
```

Returns the phase values of the equalizer spectrum flatness trace. The GUI shows only the magnitude values.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
phase: float Comma-separated list of phase values, one value per subcarrier For not allocated subcarriers, NCAP is returned. Unit: deg

6.2.18.3 Evmc

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:EVMC
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:EVMC
```

class EvmcCls

Evmc commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:EVMC
value: List[float] = driver.multiEval.trace.evmc.fetch()
```

Returns the values of the EVM vs subcarrier trace. See also 'View EVM vs Subcarrier'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ratio: float Comma-separated list of EVM values, one value per subcarrier For not allocated subcarriers, NCAP is returned. Unit: %

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:EVMC
value: List[float] = driver.multiEval.trace.evmc.read()
```

Returns the values of the EVM vs subcarrier trace. See also 'View EVM vs Subcarrier'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ratio: float Comma-separated list of EVM values, one value per subcarrier For not allocated subcarriers, NCAP is returned. Unit: %

6.2.18.4 EvmSymbol

class EvmSymbolCls

EvmSymbol 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.multiEval.trace.evmSymbol.clone()
```

Subgroups

6.2.18.4.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:EVMSymbol:AVERage
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:EVMSymbol:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:EVMSymbol:AVERage
value: List[float] = driver.multiEval.trace.evmSymbol.average.fetch()
```

Returns the values of the EVM vs modulation symbol trace. See also 'View EVM'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ratio: float Comma-separated list of EVM values, one value per modulation symbol
Unit: %

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:EVMSymbol:AVERage
value: List[float] = driver.multiEval.trace.evmSymbol.average.read()
```

Returns the values of the EVM vs modulation symbol trace. See also 'View EVM'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ratio: float Comma-separated list of EVM values, one value per modulation symbol
Unit: %

6.2.18.4.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:EVMSymbol:CURRent
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:EVMSymbol:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:EVMSymbol:CURRent
value: List[float] = driver.multiEval.trace.evmSymbol.current.fetch()
```

Returns the values of the EVM vs modulation symbol trace. See also 'View EVM'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ratio: float Comma-separated list of EVM values, one value per modulation symbol
Unit: %

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:EVMSymbol:CURRent
value: List[float] = driver.multiEval.trace.evmSymbol.current.read()
```

Returns the values of the EVM vs modulation symbol trace. See also 'View EVM'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ratio: float Comma-separated list of EVM values, one value per modulation symbol
Unit: %

6.2.18.4.3 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:EVMSymbol:MAXimum
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:EVMSymbol:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:EVMSymbol:MAXimum
value: List[float] = driver.multiEval.trace.evmSymbol.maximum.fetch()
```

Returns the values of the EVM vs modulation symbol trace. See also 'View EVM'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ratio: float Comma-separated list of EVM values, one value per modulation symbol
Unit: %

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:EVMSymbol:MAXimum
value: List[float] = driver.multiEval.trace.evmSymbol.maximum.read()
```

Returns the values of the EVM vs modulation symbol trace. See also 'View EVM'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
ratio: float Comma-separated list of EVM values, one value per modulation symbol
Unit: %

6.2.18.5 Iemissions

class IemissionsCls

Iemissions commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.18.5.1 Cc<CarrierComponent>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.multiEval.trace.iemissions.cc.repcap_carrierComponent_get()
driver.multiEval.trace.iemissions.cc.repcap_carrierComponent_set(repcap.CarrierComponent.
↪Nr1)
```

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:IEMissions:CC<Nr>
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:IEMissions:CC<Nr>
```

class CcCls

Cc commands group definition. 2 total commands, 0 Subgroups, 2 group commands Repeated Capability: CarrierComponent, default value after init: CarrierComponent.Nr1

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:IEMissions:CC<Nr>
value: List[float] = driver.multiEval.trace.iemissions.cc.
↪fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Returns the values of the inband emissions trace for carrier CC<no>. See also 'View Inband Emissions'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
power: float Comma-separated list of power values, one value per resource block Unit: dB

read(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:IEMissions:CC<Nr>
value: List[float] = driver.multiEval.trace.iemissions.cc.read(carrierComponent␣
↪= repcap.CarrierComponent.Default)
```

Returns the values of the inband emissions trace for carrier CC<no>. See also 'View Inband Emissions'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
power: float Comma-separated list of power values, one value per resource block Unit: dB

Cloning the Group

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

6.2.18.6 Iq

class IqCls

Iq 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.multiEval.trace.iq.clone()
```

Subgroups

6.2.18.6.1 High

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:IQ:HIGH
```

class HighCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Iphase: List[float]: float Normalized I amplitude
- Qphase: List[float]: float Normalized Q amplitude

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:IQ:HIGH
value: FetchStruct = driver.multiEval.trace.iq.high.fetch()
```

Returns the results in the I/Q constellation diagram for low and high EVM window position. There is one pair of values per modulation symbol. The results are returned in the following order: <Reliability>, {<IPhase>, <QPhase>}symbol 1, ..., {<IPhase>, <QPhase>}symbol n See also 'View I/Q Constellation'

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

6.2.18.6.2 Low

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:IQ:LOW
```

class LowCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Iphase: List[float]: float Normalized I amplitude
- Qphase: List[float]: float Normalized Q amplitude

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:IQ:LOW
value: FetchStruct = driver.multiEval.trace.iq.low.fetch()
```

Returns the results in the I/Q constellation diagram for low and high EVM window position. There is one pair of values per modulation symbol. The results are returned in the following order: <Reliability>, {<IPhase>, <QPhase>}symbol 1, ..., {<IPhase>, <QPhase>}symbol n See also 'View I/Q Constellation'

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

6.2.18.7 Pdynamics

class PdynamicsCls

Pdynamics 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.multiEval.trace.pdynamics.clone()
```

Subgroups

6.2.18.7.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:PDYNamics:AVERage
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:PDYNamics:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:PDYNamics:AVERage
value: List[float] = driver.multiEval.trace.pdynamics.average.fetch()
```

Return the values of the power dynamics traces. Each value is sampled with 48 Ts, corresponding to 1.5625 µs. The results of the current, average and maximum traces can be retrieved. See also 'View Power Dynamics'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
power: float 2048 power values, from -1100 µs to +2098.4375 µs relative to the start of the measure subframe. The values have a spacing of 1.5625 µs. The 705th value is at the start of the 'Measure Subframe' (0 µs) . The diagram in the GUI shows only a subsection of this trace. Unit: dBm

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:PDYNamics:AVERage
value: List[float] = driver.multiEval.trace.pdynamics.average.read()
```

Return the values of the power dynamics traces. Each value is sampled with 48 Ts, corresponding to 1.5625 µs. The results of the current, average and maximum traces can be retrieved. See also 'View Power Dynamics'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
power: float 2048 power values, from -1100 µs to +2098.4375 µs relative to the start of the measure subframe. The values have a spacing of 1.5625 µs. The 705th value is at the start of the 'Measure Subframe' (0 µs) . The diagram in the GUI shows only a subsection of this trace. Unit: dBm

6.2.18.7.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:PDYNamics:CURRent
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:PDYNamics:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:PDYNamics:CURRent
value: List[float] = driver.multiEval.trace.pdynamics.current.fetch()
```

Return the values of the power dynamics traces. Each value is sampled with 48 Ts, corresponding to 1.5625 µs. The results of the current, average and maximum traces can be retrieved. See also 'View Power Dynamics'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
power: float 2048 power values, from -1100 µs to +2098.4375 µs relative to the start of the measure subframe. The values have a spacing of 1.5625 µs. The 705th value is at the start of the 'Measure Subframe' (0 µs) . The diagram in the GUI shows only a subsection of this trace. Unit: dBm

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:PDYNamics:CURRent
value: List[float] = driver.multiEval.trace.pdynamics.current.read()
```

Return the values of the power dynamics traces. Each value is sampled with 48 Ts, corresponding to 1.5625 µs. The results of the current, average and maximum traces can be retrieved. See also 'View Power Dynamics'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
power: float 2048 power values, from -1100 µs to +2098.4375 µs relative to the start of the measure subframe. The values have a spacing of 1.5625 µs. The 705th value is at the start of the 'Measure Subframe' (0 µs) . The diagram in the GUI shows only a subsection of this trace. Unit: dBm

6.2.18.7.3 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:PDYNamics:MAXimum
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:PDYNamics:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:PDYNamics:MAXimum
value: List[float] = driver.multiEval.trace.pdynamics.maximum.fetch()
```

Return the values of the power dynamics traces. Each value is sampled with 48 Ts, corresponding to 1.5625 µs. The results of the current, average and maximum traces can be retrieved. See also 'View Power Dynamics'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
power: float 2048 power values, from -1100 µs to +2098.4375 µs relative to the start of the measure subframe. The values have a spacing of 1.5625 µs. The 705th value is at the start of the 'Measure Subframe' (0 µs) . The diagram in the GUI shows only a subsection of this trace. Unit: dBm

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:PDYNamics:MAXimum
value: List[float] = driver.multiEval.trace.pdynamics.maximum.read()
```

Return the values of the power dynamics traces. Each value is sampled with 48 Ts, corresponding to 1.5625 µs. The results of the current, average and maximum traces can be retrieved. See also 'View Power Dynamics'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
power: float 2048 power values, from -1100 µs to +2098.4375 µs relative to the start of the measure subframe. The values have a spacing of 1.5625 µs. The 705th value is at the start of the 'Measure Subframe' (0 µs) . The diagram in the GUI shows only a subsection of this trace. Unit: dBm

6.2.18.8 Pmonitor

class PmonitorCls

Pmonitor commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.18.8.1 Cc<CarrierComponent>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.multiEval.trace.pmonitor.cc.repcap_carrierComponent_get()
driver.multiEval.trace.pmonitor.cc.repcap_carrierComponent_set(repcap.CarrierComponent.
↪Nr1)
```

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:PMONitor:CC<Nr>
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:PMONitor:CC<Nr>
```

class CcCls

Cc commands group definition. 2 total commands, 0 Subgroups, 2 group commands Repeated Capability: CarrierComponent, default value after init: CarrierComponent.Nr1

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:PMONitor:CC<Nr>
value: List[float] = driver.multiEval.trace.pmonitor.cc.fetch(carrierComponent␣
↪= repcap.CarrierComponent.Default)
```

Returns the power monitor results for all captured CC<no> subframes.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
power: float Comma-separated list of power values, one value per subframe Unit: dBm

read(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:PMONitor:CC<Nr>
value: List[float] = driver.multiEval.trace.pmonitor.cc.read(carrierComponent =␣
↪repcap.CarrierComponent.Default)
```

Returns the power monitor results for all captured CC<no> subframes.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
power: float Comma-separated list of power values, one value per subframe Unit: dBm

Cloning the Group

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

6.2.18.9 RbaTable

class RbaTableCls

RbaTable commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.18.9.1 Cc<CarrierComponent>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.multiEval.trace.rbaTable.cc.repcap_carrierComponent_get()
driver.multiEval.trace.rbaTable.cc.repcap_carrierComponent_set(repcap.CarrierComponent.
↪Nr1)
```

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:RBATable:CC<Nr>
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:RBATable:CC<Nr>
```

class CcCls

Cc commands group definition. 2 total commands, 0 Subgroups, 2 group commands Repeated Capability: CarrierComponent, default value after init: CarrierComponent.Nr1

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Channel_Type: List[enums.RbTableChannelType]: PUSCh | PUCCh | NONE | DL | SSUB | PSSCh | PSCCh PUSCh / PUCCH: for UL slot with RB allocation PSSCh / PSCCh: for SL subframe with RB allocation NONE: UL slot or SL subframe contains no allocated RBs. DL: DL slot (only for TDD UL measurements) SSUB: part of special SF (only for TDD UL measurements)
- Offset_Rb: List[int]: decimal Offset of first allocated RB for the given channel type
- No_Rb: List[int]: decimal Number of allocated RBs for the given channel type

fetch(*carrierComponent=CarrierComponent.Default*) → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:RBATable:CC<Nr>
value: ResultData = driver.multiEval.trace.rbaTable.cc.fetch(carrierComponent =␣
↪repcap.CarrierComponent.Default)
```

Returns the information of the CC<no> RB allocation table. See also 'View RB Allocation Table'. For uplink measurements, there are three results per captured slot (n = number of captured subframes) : <Reliability>, {<ChannelType>, <OffsetRB>, <NoRB>}slot 1, ..., {...}slot (n*2) For sidelink measurements, there are six results per captured subframe (SF) , three for the PSCCH and three for the PSSCH: <Reliability>, {...}SF 1 (PSCCH) , {...}SF 1 (PSSCH) , ..., {...}SF n (PSCCH) , {.. .}SF n (PSSCH)

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

read(*carrierComponent=CarrierComponent.Default*) → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:RBATable:CC<Nr>
value: ResultData = driver.multiEval.trace.rbaTable.cc.read(carrierComponent =␣
↪repcap.CarrierComponent.Default)
```

Returns the information of the CC<no> RB allocation table. See also 'View RB Allocation Table'. For uplink measurements, there are three results per captured slot (n = number of captured subframes) : <Reliability>, {<ChannelType>, <OffsetRB>, <NoRB>}slot 1, ..., {...}slot (n*2) For sidelink measurements, there are six results per captured subframe (SF) , three for the PSCCH and three for the PSSCH: <Reliability>, {...}SF 1 (PSCCH) , {...}SF 1 (PSSCH) , ..., {...}SF n (PSCCH) , {.. .}SF n (PSSCH)

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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.multiEval.trace.rbaTable.cc.clone()
```

6.2.18.10 SeMask

class SeMaskCls

SeMask 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.multiEval.trace.seMask.clone()
```

Subgroups

6.2.18.10.1 Rbw<RBWkHz>

RepCap Settings

```
# Range: Rbw30 .. Rbw1000
rc = driver.multiEval.trace.seMask.rbw.repcap_rBWkHz_get()
driver.multiEval.trace.seMask.rbw.repcap_rBWkHz_set(repcap.RBWkHz.Rbw30)
```

class RbwCls

Rbw commands group definition. 6 total commands, 3 Subgroups, 0 group commands Repeated Capability: RBWkHz, default value after init: RBWkHz.Rbw30

Cloning the Group

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

Subgroups

6.2.18.10.1.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:SEMask:RBW<kHz>:AVERage
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:SEMask:RBW<kHz>:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*rBWkHz=RBWkHz.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:SEMask:RBW<kHz>
↪:AVERage
value: List[float] = driver.multiEval.trace.seMask.rbw.average.fetch(rBWkHz =␣
↪repcap.RBWkHz.Default)
```

Returns the values of the spectrum emission traces. Separate traces are available for the individual resolution bandwidths (<kHz>) . The results of the current, average and maximum traces can be retrieved. See also 'View Spectrum Emission Mask'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param rBWkHz
optional repeated capability selector. Default value: Rbw30 (settable in the interface 'Rbw')

return
power: float Comma-separated list of power results The value in the middle of the result array corresponds to the center frequency. The test point separation between two results depends on the resolution bandwidth, see table below. For RBW100 and greater, results are only available for frequencies with active limits using these RBWs. For other frequencies, INV is returned. Unit: dBm

read(*rBWkHz=RBWkHz.Default*) → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:SEMask:RBW<kHz>:AVERage
value: List[float] = driver.multiEval.trace.seMask.rbw.average.read(rBWkHz =␣
↪repcap.RBWkHz.Default)
```

Returns the values of the spectrum emission traces. Separate traces are available for the individual resolution bandwidths (<kHz>) . The results of the current, average and maximum traces can be retrieved. See also 'View Spectrum Emission Mask'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param rBWkHz
optional repeated capability selector. Default value: Rbw30 (settable in the interface 'Rbw')

return
power: float Comma-separated list of power results The value in the middle of the result array corresponds to the center frequency. The test point separation between two results depends on the resolution bandwidth, see table below. For RBW100 and greater, results are only available for frequencies with active limits using these RBWs. For other frequencies, INV is returned. Unit: dBm

6.2.18.10.1.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:SEMask:RBW<kHz>:CURRent
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:SEMask:RBW<kHz>:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*rBWkHz=RBWkHz.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:SEMask:RBW<kHz>
↪:CURRent
value: List[float] = driver.multiEval.trace.seMask.rbw.current.fetch(rBWkHz =␣
↪repcap.RBWkHz.Default)
```

Returns the values of the spectrum emission traces. Separate traces are available for the individual resolution bandwidths (<kHz>) . The results of the current, average and maximum traces can be retrieved. See also 'View Spectrum Emission Mask'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param rBWkHz
optional repeated capability selector. Default value: Rbw30 (settable in the interface 'Rbw')

return
power: float Comma-separated list of power results The value in the middle of the result array corresponds to the center frequency. The test point separation between two results depends on the resolution bandwidth, see table below. For RBW100 and greater, results are only available for frequencies with active limits using these RBWs. For other frequencies, INV is returned. Unit: dBm

read(*rBWkHz=RBWkHz.Default*) → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:SEMask:RBW<kHz>:CURRent
value: List[float] = driver.multiEval.trace.seMask.rbw.current.read(rBWkHz =␣
↪repcap.RBWkHz.Default)
```

Returns the values of the spectrum emission traces. Separate traces are available for the individual resolution bandwidths (<kHz>) . The results of the current, average and maximum traces can be retrieved. See also 'View Spectrum Emission Mask'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param rBWkHz
optional repeated capability selector. Default value: Rbw30 (settable in the interface 'Rbw')

return
power: float Comma-separated list of power results The value in the middle of the result array corresponds to the center frequency. The test point separation between two results depends on the resolution bandwidth, see table below. For RBW100 and greater, results are only available for frequencies with active limits using these RBWs. For other frequencies, INV is returned. Unit: dBm

6.2.18.10.1.3 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:SEMask:RBW<kHz>:MAXimum
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:SEMask:RBW<kHz>:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*rBWkHz=RBWkHz.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:SEMask:RBW<kHz>
↪:MAXimum
value: List[float] = driver.multiEval.trace.seMask.rbw.maximum.fetch(rBWkHz =␣
↪repcap.RBWkHz.Default)
```

Returns the values of the spectrum emission traces. Separate traces are available for the individual resolution bandwidths (<kHz>) . The results of the current, average and maximum traces can be retrieved. See also 'View Spectrum Emission Mask'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param rBWkHz
optional repeated capability selector. Default value: Rbw30 (settable in the interface 'Rbw')

return
power: float Comma-separated list of power results The value in the middle of the result array corresponds to the center frequency. The test point separation between two results depends on the resolution bandwidth, see table below. For RBW100 and greater, results are only available for frequencies with active limits using these RBWs. For other frequencies, INV is returned. Unit: dBm

read(*rBWkHz=RBWkHz.Default*) → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:SEMask:RBW<kHz>:MAXimum
value: List[float] = driver.multiEval.trace.seMask.rbw.maximum.read(rBWkHz =␣
↪repcap.RBWkHz.Default)
```

Returns the values of the spectrum emission traces. Separate traces are available for the individual resolution bandwidths (<kHz>) . The results of the current, average and maximum traces can be retrieved. See also 'View Spectrum Emission Mask'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param rBWkHz
optional repeated capability selector. Default value: Rbw30 (settable in the interface 'Rbw')

return
power: float Comma-separated list of power results The value in the middle of the result array corresponds to the center frequency. The test point separation between two results depends on the resolution bandwidth, see table below. For RBW100 and greater, results are only available for frequencies with active limits using these RBWs. For other frequencies, INV is returned. Unit: dBm

6.2.19 VfThroughput

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:VFTHroughput
```

class VfThroughputCls

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

fetch() → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:VFTHroughput
value: float = driver.multiEval.vfThroughput.fetch()
```

Queries the 'View Filter Throughput'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
vf_throughput: float Unit: %

6.3 Prach

SCPI Commands :

```
INITiate:LTE:MEASurement<Instance>:PRACh
STOP:LTE:MEASurement<Instance>:PRACh
ABORt:LTE:MEASurement<Instance>:PRACh
```

class PrachCls

Prach commands group definition. 80 total commands, 5 Subgroups, 3 group commands

abort(*opc_timeout_ms: int = -1*) → None

```
# SCPI: ABORt:LTE:MEASurement<Instance>:PRACh
driver.prach.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
↪' 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:LTE:MEASurement<Instance>:PRACh
driver.prach.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() → None

```
# SCPI: STOP:LTE:MEASurement<Instance>:PRACh
driver.prach.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.

stop_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: STOP:LTE:MEASurement<Instance>:PRACh
driver.prach.stop_with_opc()


    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.

Same as stop, but waits for the operation to complete before continuing further. Use the RsCmwLteMeas.utilities.opc_timeout_set() to set the timeout value.

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.prach.clone()
```

Subgroups

6.3.1 EvmSymbol

class EvmSymbolCls

EvmSymbol 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.prach.evmSymbol.clone()
```

Subgroups

6.3.1.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:EVMSymbol:AVERage
FETCh:LTE:MEASurement<Instance>:PRACh:EVMSymbol:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Low: List[float]: float EVM value for low EVM window position. Unit: %
- High: List[float]: float EVM value for high EVM window position. Unit: %

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:EVMSymbol:AVERage
value: ResultData = driver.prach.evmSymbol.average.fetch()
```

Returns the values of the EVM RMS bar graphs for the OFDM symbols in the measured preamble. The results of the current, average and maximum bar graphs can be retrieved. There is one pair of EVM values per OFDM symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1 If the preamble contains only one symbol, NCAPs are returned for the remaining symbol. See also 'View EVM vs Symbol'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:PRACh:EVMSymbol:AVERage
value: ResultData = driver.prach.evmSymbol.average.read()
```

Returns the values of the EVM RMS bar graphs for the OFDM symbols in the measured preamble. The results of the current, average and maximum bar graphs can be retrieved. There is one pair of EVM values per OFDM symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1 If the preamble contains only one symbol, NCAPs are returned for the remaining symbol. See also 'View EVM vs Symbol'.

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

6.3.1.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:EVMSymbol:CURRent
FETCh:LTE:MEASurement<Instance>:PRACh:EVMSymbol:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Low: List[float]: float EVM value for low EVM window position. Unit: %
- High: List[float]: float EVM value for high EVM window position. Unit: %

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:EVMSymbol:CURRent
value: ResultData = driver.prach.evmSymbol.current.fetch()
```

Returns the values of the EVM RMS bar graphs for the OFDM symbols in the measured preamble. The results of the current, average and maximum bar graphs can be retrieved. There is one pair of EVM values per OFDM symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1 If the preamble contains only one symbol, NCAPs are returned for the remaining symbol. See also 'View EVM vs Symbol'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:PRACh:EVMSymbol:CURRent
value: ResultData = driver.prach.evmSymbol.current.read()
```

Returns the values of the EVM RMS bar graphs for the OFDM symbols in the measured preamble. The results of the current, average and maximum bar graphs can be retrieved. There is one pair of EVM values per OFDM symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1 If the preamble contains only one symbol, NCAPs are returned for the remaining symbol. See also 'View EVM vs Symbol'.

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

6.3.1.3 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:EVMSymbol:MAXimum
FETCh:LTE:MEASurement<Instance>:PRACh:EVMSymbol:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Low: List[float]: float EVM value for low EVM window position. Unit: %
- High: List[float]: float EVM value for high EVM window position. Unit: %

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:EVMSymbol:MAXimum
value: ResultData = driver.prach.evmSymbol.maximum.fetch()
```

Returns the values of the EVM RMS bar graphs for the OFDM symbols in the measured preamble. The results of the current, average and maximum bar graphs can be retrieved. There is one pair of EVM values per OFDM symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1 If the preamble contains only one symbol, NCAPs are returned for the remaining symbol. See also 'View EVM vs Symbol'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:PRACh:EVMSymbol:MAXimum
value: ResultData = driver.prach.evmSymbol.maximum.read()
```

Returns the values of the EVM RMS bar graphs for the OFDM symbols in the measured preamble. The results of the current, average and maximum bar graphs can be retrieved. There is one pair of EVM values per OFDM symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1 If the preamble contains only one symbol, NCAPs are returned for the remaining symbol. See also 'View EVM vs Symbol'.

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

6.3.1.4 Peak

class PeakCls

Peak 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.prach.evmSymbol.peak.clone()
```

Subgroups

6.3.1.4.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:EVMSymbol:PEAK:AVERage
FETCh:LTE:MEASurement<Instance>:PRACh:EVMSymbol:PEAK:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Low: List[float]: float EVM value for low EVM window position. Unit: %
- High: List[float]: float EVM value for high EVM window position. Unit: %

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:EVMSymbol:PEAK:AVERage
value: ResultData = driver.prach.evmSymbol.peak.average.fetch()
```

Returns the values of the EVM peak bar graphs for the OFDM symbols in the measured preamble. The results of the current, average and maximum bar graphs can be retrieved. There is one pair of EVM values per OFDM symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1 If the preamble contains only one symbol, NCAPs are returned for the remaining symbol. See also 'View EVM vs Symbol'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:PRACh:EVMSymbol:PEAK:AVERage
value: ResultData = driver.prach.evmSymbol.peak.average.read()
```

Returns the values of the EVM peak bar graphs for the OFDM symbols in the measured preamble. The results of the current, average and maximum bar graphs can be retrieved. There is one pair of EVM values per OFDM symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>,

<High>}symbol 1 If the preamble contains only one symbol, NCAPs are returned for the remaining symbol. See also 'View EVM vs Symbol'.

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

6.3.1.4.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:EVMSymbol:PEAK:CURRent
FETCh:LTE:MEASurement<Instance>:PRACh:EVMSymbol:PEAK:CURRent
```

`class CurrentCls`

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

`class ResultData`

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Low: List[float]: float EVM value for low EVM window position. Unit: %
- High: List[float]: float EVM value for high EVM window position. Unit: %

`fetch()` → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:EVMSymbol:PEAK:CURRent
value: ResultData = driver.prach.evmSymbol.peak.current.fetch()
```

Returns the values of the EVM peak bar graphs for the OFDM symbols in the measured preamble. The results of the current, average and maximum bar graphs can be retrieved. There is one pair of EVM values per OFDM symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1 If the preamble contains only one symbol, NCAPs are returned for the remaining symbol. See also 'View EVM vs Symbol'.

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

`read()` → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:PRACh:EVMSymbol:PEAK:CURRent
value: ResultData = driver.prach.evmSymbol.peak.current.read()
```

Returns the values of the EVM peak bar graphs for the OFDM symbols in the measured preamble. The results of the current, average and maximum bar graphs can be retrieved. There is one pair of EVM values per OFDM symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1 If the preamble contains only one symbol, NCAPs are returned for the remaining symbol. See also 'View EVM vs Symbol'.

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

6.3.1.4.3 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:EVMSymbol:PEAK:MAXimum
FETCh:LTE:MEASurement<Instance>:PRACh:EVMSymbol:PEAK:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Low: List[float]: float EVM value for low EVM window position. Unit: %
- High: List[float]: float EVM value for high EVM window position. Unit: %

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:EVMSymbol:PEAK:MAXimum
value: ResultData = driver.prach.evmSymbol.peak.maximum.fetch()
```

Returns the values of the EVM peak bar graphs for the OFDM symbols in the measured preamble. The results of the current, average and maximum bar graphs can be retrieved. There is one pair of EVM values per OFDM symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1 If the preamble contains only one symbol, NCAPs are returned for the remaining symbol. See also 'View EVM vs Symbol'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:PRACh:EVMSymbol:PEAK:MAXimum
value: ResultData = driver.prach.evmSymbol.peak.maximum.read()
```

Returns the values of the EVM peak bar graphs for the OFDM symbols in the measured preamble. The results of the current, average and maximum bar graphs can be retrieved. There is one pair of EVM values per OFDM symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1 If the preamble contains only one symbol, NCAPs are returned for the remaining symbol. See also 'View EVM vs Symbol'.

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

6.3.2 Modulation

class ModulationCls

Modulation commands group definition. 20 total commands, 9 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.2.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:MODulation:AVERage
FETCh:LTE:MEASurement<Instance>:PRACh:MODulation:AVERage
CALCulate:LTE:MEASurement<Instance>:PRACh:MODulation:AVERage
```

class AverageCls

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

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits. Unit: %
- Evm_Rms_Low: float or bool: float EVM RMS value, low EVM window position. Unit: %
- Evm_Rms_High: float or bool: float EVM RMS value, high EVM window position. Unit: %
- Evm_Peak_Low: float or bool: float EVM peak value, low EVM window position. Unit: %
- Evm_Peak_High: float or bool: float EVM peak value, high EVM window position. Unit: %
- Mag_Error_Rms_Low: float or bool: float Magnitude error RMS value, low EVM window position. Unit: %
- Mag_Error_Rms_High: float or bool: float Magnitude error RMS value, low EVM window position. Unit: %
- Mag_Error_Peak_Low: float or bool: float Magnitude error peak value, low EVM window position. Unit: %
- Mag_Err_Peak_High: float or bool: float Magnitude error peak value, high EVM window position. Unit: %
- Ph_Error_Rms_Low: float or bool: float Phase error RMS value, low EVM window position. Unit: deg
- Ph_Error_Rms_High: float or bool: float Phase error RMS value, high EVM window position. Unit: deg

- Ph_Error_Peak_Low: float or bool: float Phase error peak value, low EVM window position. Unit: deg
- Ph_Error_Peak_High: float or bool: float Phase error peak value, high EVM window position. Unit: deg
- Frequency_Error: float or bool: float Carrier frequency error. Unit: Hz
- Timing_Error: float or bool: float Transmit time error. Unit: Ts (basic LTE time unit)
- Tx_Power: float or bool: float User equipment power. Unit: dBm
- Peak_Power: float or bool: float User equipment peak power. Unit: dBm

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits. Unit: %
- Evm_Rms_Low: float: float EVM RMS value, low EVM window position. Unit: %
- Evm_Rms_High: float: float EVM RMS value, high EVM window position. Unit: %
- Evm_Peak_Low: float: float EVM peak value, low EVM window position. Unit: %
- Evm_Peak_High: float: float EVM peak value, high EVM window position. Unit: %
- Mag_Error_Rms_Low: float: float Magnitude error RMS value, low EVM window position. Unit: %
- Mag_Error_Rms_High: float: float Magnitude error RMS value, low EVM window position. Unit: %
- Mag_Error_Peak_Low: float: float Magnitude error peak value, low EVM window position. Unit: %
- Mag_Err_Peak_High: float: float Magnitude error peak value, high EVM window position. Unit: %
- Ph_Error_Rms_Low: float: float Phase error RMS value, low EVM window position. Unit: deg
- Ph_Error_Rms_High: float: float Phase error RMS value, high EVM window position. Unit: deg
- Ph_Error_Peak_Low: float: float Phase error peak value, low EVM window position. Unit: deg
- Ph_Error_Peak_High: float: float Phase error peak value, high EVM window position. Unit: deg
- Frequency_Error: float: float Carrier frequency error. Unit: Hz
- Timing_Error: float: float Transmit time error. Unit: Ts (basic LTE time unit)
- Tx_Power: float: float User equipment power. Unit: dBm
- Peak_Power: float: float User equipment peak power. Unit: dBm

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:PRACh:MODulation:AVERage
value: CalculateStruct = driver.prach.modulation.average.calculate()
```

Return the current, average and standard deviation single-value 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:LTE:MEASurement<Instance>:PRACh:MODulation:AVERage
value: ResultData = driver.prach.modulation.average.fetch()
```

Return the current, average and standard deviation single-value 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:LTE:MEASurement<Instance>:PRACh:MODulation:AVERage
value: ResultData = driver.prach.modulation.average.read()
```

Return the current, average and standard deviation single-value 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.

6.3.2.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:MODulation:CURRent
FETCh:LTE:MEASurement<Instance>:PRACh:MODulation:CURRent
CALCulate:LTE:MEASurement<Instance>:PRACh:MODulation:CURRent
```

class CurrentCls

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

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits. Unit: %
- Evm_Rms_Low: float or bool: float EVM RMS value, low EVM window position. Unit: %
- Evm_Rms_High: float or bool: float EVM RMS value, high EVM window position. Unit: %
- Evm_Peak_Low: float or bool: float EVM peak value, low EVM window position. Unit: %
- Evm_Peak_High: float or bool: float EVM peak value, high EVM window position. Unit: %
- Mag_Error_Rms_Low: float or bool: float Magnitude error RMS value, low EVM window position. Unit: %
- Mag_Error_Rms_High: float or bool: float Magnitude error RMS value, low EVM window position. Unit: %
- Mag_Error_Peak_Low: float or bool: float Magnitude error peak value, low EVM window position. Unit: %

- Mag_Err_Peak_High: float or bool: float Magnitude error peak value, high EVM window position. Unit: %
- Ph_Error_Rms_Low: float or bool: float Phase error RMS value, low EVM window position. Unit: deg
- Ph_Error_Rms_High: float or bool: float Phase error RMS value, high EVM window position. Unit: deg
- Ph_Error_Peak_Low: float or bool: float Phase error peak value, low EVM window position. Unit: deg
- Ph_Error_Peak_High: float or bool: float Phase error peak value, high EVM window position. Unit: deg
- Frequency_Error: float or bool: float Carrier frequency error. Unit: Hz
- Timing_Error: float or bool: float Transmit time error. Unit: Ts (basic LTE time unit)
- Tx_Power: float or bool: float User equipment power. Unit: dBm
- Peak_Power: float or bool: float User equipment peak power. Unit: dBm

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits. Unit: %
- Evm_Rms_Low: float: float EVM RMS value, low EVM window position. Unit: %
- Evm_Rms_High: float: float EVM RMS value, high EVM window position. Unit: %
- Evm_Peak_Low: float: float EVM peak value, low EVM window position. Unit: %
- Evm_Peak_High: float: float EVM peak value, high EVM window position. Unit: %
- Mag_Error_Rms_Low: float: float Magnitude error RMS value, low EVM window position. Unit: %
- Mag_Error_Rms_High: float: float Magnitude error RMS value, low EVM window position. Unit: %
- Mag_Error_Peak_Low: float: float Magnitude error peak value, low EVM window position. Unit: %
- Mag_Err_Peak_High: float: float Magnitude error peak value, high EVM window position. Unit: %
- Ph_Error_Rms_Low: float: float Phase error RMS value, low EVM window position. Unit: deg
- Ph_Error_Rms_High: float: float Phase error RMS value, high EVM window position. Unit: deg
- Ph_Error_Peak_Low: float: float Phase error peak value, low EVM window position. Unit: deg
- Ph_Error_Peak_High: float: float Phase error peak value, high EVM window position. Unit: deg
- Frequency_Error: float: float Carrier frequency error. Unit: Hz
- Timing_Error: float: float Transmit time error. Unit: Ts (basic LTE time unit)
- Tx_Power: float: float User equipment power. Unit: dBm
- Peak_Power: float: float User equipment peak power. Unit: dBm

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:PRACh:MODulation:CURRent
value: CalculateStruct = driver.prach.modulation.current.calculate()
```

Return the current, average and standard deviation single-value 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:LTE:MEASurement<Instance>:PRACh:MODulation:CURRent
value: ResultData = driver.prach.modulation.current.fetch()
```

Return the current, average and standard deviation single-value 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:LTE:MEASurement<Instance>:PRACh:MODulation:CURRent
value: ResultData = driver.prach.modulation.current.read()
```

Return the current, average and standard deviation single-value 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.

6.3.2.3 DpfOffset

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:PRACh:MODulation:DPFoffset
```

class DpfOffsetCls

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

fetch() → int

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:MODulation:DPFoffset
value: int = driver.prach.modulation.dpfOffset.fetch()
```

Returns the automatically detected or manually configured PRACH frequency offset for single-preamble measurements.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
prach_freq_offset: decimal PRACH frequency offset

Cloning the Group

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

Subgroups

6.3.2.3.1 Preamble<Preamble>

RepCap Settings

```
# Range: Nr1 .. Nr400
rc = driver.prach.modulation.dpfOffset.preamble.repcap_preamble_get()
driver.prach.modulation.dpfOffset.preamble.repcap_preamble_set(repcap.Preamble.Nr1)
```

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:PRACh:MODulation:DPFoffset:PREamble<Number>
```

class PreambleCls

Preamble commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Preamble, default value after init: Preamble.Nr1

fetch(*preamble=Preamble.Default*) → int

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:MODulation:DPFoffset:PREamble
↪<Number>
value: int = driver.prach.modulation.dpfOffset.preamble.fetch(preamble = repcap.
↪Preamble.Default)
```

Returns the automatically detected or manually configured PRACH frequency offset for a selected preamble of multi-preamble measurements.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param preamble
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Preamble')

return
prach_freq_offset: decimal PRACH frequency offset

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.prach.modulation.dpfOffset.preamble.clone()
```

6.3.2.4 DsIndex

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:PRACh:MODulation:DSINdex
```

class DsIndexCls

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

fetch() → int

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:MODulation:DSINdex
value: int = driver.prach.modulation.dsIndex.fetch()
```

Returns the automatically detected or manually configured sequence index for single-preamble measurements.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
sequence_index: decimal Sequence index

Cloning the Group

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

Subgroups

6.3.2.4.1 Preamble<Preamble>

RepCap Settings

```
# Range: Nr1 .. Nr400
rc = driver.prach.modulation.dsIndex.preamble.repcap_preamble_get()
driver.prach.modulation.dsIndex.preamble.repcap_preamble_set(repcap.Preamble.Nr1)
```

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:PRACh:MODulation:DSINdex:PREamble<Number>
```

class PreambleCls

Preamble commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Preamble, default value after init: Preamble.Nr1

fetch(*preamble=Preamble.Default*) → int

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:MODulation:DSINdex:PREamble
↪<Number>
value: int = driver.prach.modulation.dsIndex.preamble.fetch(preamble = repcap.
↪Preamble.Default)
```

Returns the automatically detected or manually configured sequence index for a selected preamble of multi-preamble measurements.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param preamble
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Preamble')

return
sequence_index: decimal Sequence index

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.prach.modulation.dsIndex.preamble.clone()
```

6.3.2.5 Extreme

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:MODulation:EXTReme
FETCh:LTE:MEASurement<Instance>:PRACh:MODulation:EXTReme
CALCulate:LTE:MEASurement<Instance>:PRACh:MODulation:EXTReme
```

class ExtremeCls

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

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits. Unit: %
- Evm_Rms_Low: float or bool: float EVM RMS value, low EVM window position Unit: %
- Evm_Rms_High: float or bool: float EVM RMS value, high EVM window position Unit: %

- Evm_Peak_Low: float or bool: float EVM peak value, low EVM window position Unit: %
- Evm_Peak_High: float or bool: float EVM peak value, high EVM window position Unit: %
- Mag_Error_Rms_Low: float or bool: float Magnitude error RMS value, low EVM window position Unit: %
- Mag_Error_Rms_High: float or bool: float Magnitude error RMS value, low EVM window position Unit: %
- Mag_Error_Peak_Low: float or bool: float Magnitude error peak value, low EVM window position Unit: %
- Mag_Err_Peak_High: float or bool: float Magnitude error peak value, high EVM window position Unit: %
- Ph_Error_Rms_Low: float or bool: float Phase error RMS value, low EVM window position Unit: deg
- Ph_Error_Rms_High: float or bool: float Phase error RMS value, high EVM window position Unit: deg
- Ph_Error_Peak_Low: float or bool: float Phase error peak value, low EVM window position Unit: deg
- Ph_Error_Peak_High: float or bool: float Phase error peak value, high EVM window position Unit: deg
- Frequency_Error: float or bool: float Carrier frequency error Unit: Hz
- Timing_Error: float or bool: float Time error Unit: Ts (basic LTE time unit)
- Tx_Power_Minimum: float or bool: float Minimum user equipment power Unit: dBm
- Tx_Power_Maximum: float or bool: float Maximum user equipment power Unit: dBm
- Peak_Power_Min: float or bool: float Minimum user equipment peak power Unit: dBm
- Peak_Power_Max: float or bool: float Maximum user equipment peak power Unit: dBm

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits. Unit: %
- Evm_Rms_Low: float: float EVM RMS value, low EVM window position Unit: %
- Evm_Rms_High: float: float EVM RMS value, high EVM window position Unit: %
- Evm_Peak_Low: float: float EVM peak value, low EVM window position Unit: %
- Evm_Peak_High: float: float EVM peak value, high EVM window position Unit: %
- Mag_Error_Rms_Low: float: float Magnitude error RMS value, low EVM window position Unit: %
- Mag_Error_Rms_High: float: float Magnitude error RMS value, low EVM window position Unit: %
- Mag_Error_Peak_Low: float: float Magnitude error peak value, low EVM window position Unit: %
- Mag_Err_Peak_High: float: float Magnitude error peak value, high EVM window position Unit: %
- Ph_Error_Rms_Low: float: float Phase error RMS value, low EVM window position Unit: deg
- Ph_Error_Rms_High: float: float Phase error RMS value, high EVM window position Unit: deg
- Ph_Error_Peak_Low: float: float Phase error peak value, low EVM window position Unit: deg

- Ph_Error_Peak_High: float: float Phase error peak value, high EVM window position Unit: deg
- Frequency_Error: float: float Carrier frequency error Unit: Hz
- Timing_Error: float: float Time error Unit: Ts (basic LTE time unit)
- Tx_Power_Minimum: float: float Minimum user equipment power Unit: dBm
- Tx_Power_Maximum: float: float Maximum user equipment power Unit: dBm
- Peak_Power_Min: float: float Minimum user equipment peak power Unit: dBm
- Peak_Power_Max: float: float Maximum user equipment peak power Unit: dBm

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:PRACh:MODulation:EXTReme
value: CalculateStruct = driver.prach.modulation.extreme.calculate()
```

Returns the extreme single value 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:LTE:MEASurement<Instance>:PRACh:MODulation:EXTReme
value: ResultData = driver.prach.modulation.extreme.fetch()
```

Returns the extreme single value 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:LTE:MEASurement<Instance>:PRACh:MODulation:EXTReme
value: ResultData = driver.prach.modulation.extreme.read()
```

Returns the extreme single value 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.

6.3.2.6 Nsymbol

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:PRACh:MODulation:NSYMbol
```

class NsymbolCls

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

fetch() → int

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:MODulation:NSYMbol
value: int = driver.prach.modulation.nsymbol.fetch()
```

Queries the number of active OFDM symbols (symbols with result bars) in the EVM vs symbol bar graph.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
no_of_symbols: decimal

6.3.2.7 Preamble<Preamble>

RepCap Settings

```
# Range: Nr1 .. Nr400
rc = driver.prach.modulation.preamble.repcap_preamble_get()
driver.prach.modulation.preamble.repcap_preamble_set(repcap.Preamble.Nr1)
```

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:MODulation:PREamble<Number>
FETCh:LTE:MEASurement<Instance>:PRACh:MODulation:PREamble<Number>
```

class PreambleCls

Preamble commands group definition. 2 total commands, 0 Subgroups, 2 group commands Repeated Capability: Preamble, default value after init: Preamble.Nr1

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Preamble_Rel: int: decimal Reliability indicator for the preamble.
- Evm_Rms_Low: float: float EVM RMS value, low EVM window position. Unit: %
- Evm_Rms_High: float: float EVM RMS value, high EVM window position. Unit: %
- Evm_Peak_Low: float: float EVM peak value, low EVM window position. Unit: %
- Evm_Peak_High: float: float EVM peak value, high EVM window position. Unit: %
- Mag_Error_Rms_Low: float: float Magnitude error RMS value, low EVM window position. Unit: %
- Mag_Error_Rms_High: float: float Magnitude error RMS value, low EVM window position. Unit: %
- Mag_Error_Peak_Low: float: float Magnitude error peak value, low EVM window position. Unit: %
- Mag_Err_Peak_High: float: float Magnitude error peak value, high EVM window position. Unit: %
- Ph_Error_Rms_Low: float: float Phase error RMS value, low EVM window position. Unit: deg
- Ph_Error_Rms_High: float: float Phase error RMS value, high EVM window position. Unit: deg
- Ph_Error_Peak_Low: float: float Phase error peak value, low EVM window position. Unit: deg
- Ph_Error_Peak_High: float: float Phase error peak value, high EVM window position. Unit: deg

- Frequency_Error: float: float Carrier frequency error. Unit: Hz
- Timing_Error: float: float Transmit time error. Unit: Ts (basic LTE time unit)
- Tx_Power: float: float User equipment power. Unit: dBm
- Peak_Power: float: float User equipment peak power. Unit: dBm

fetch(*preamble=Preamble.Default*) → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:MODulation:PREamble<Number>
value: ResultData = driver.prach.modulation.preamble.fetch(preamble = repcap.
↪Preamble.Default)
```

Return the single value results of the 'EVM vs Preamble' and 'Power vs Preamble' views, for a selected preamble. See also 'View EVM vs Preamble, Power vs Preamble'.

param preamble
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Preamble')

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

read(*preamble=Preamble.Default*) → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:PRACh:MODulation:PREamble<Number>
value: ResultData = driver.prach.modulation.preamble.read(preamble = repcap.
↪Preamble.Default)
```

Return the single value results of the 'EVM vs Preamble' and 'Power vs Preamble' views, for a selected preamble. See also 'View EVM vs Preamble, Power vs Preamble'.

param preamble
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Preamble')

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.prach.modulation.preamble.clone()
```

6.3.2.8 Scorrelation

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:PRACh:MODulation:SCORrelation
```

class ScorrelationCls

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

fetch() → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:MODulation:SCORrelation
value: float = driver.prach.modulation.scorrelation.fetch()
```

Returns the sequence correlation for single-preamble measurements. It indicates the correlation between the ideal preamble sequence determined from the parameter settings and the measured preamble sequence. A value of 1 corresponds to perfect correlation. A value much smaller than 1 indicates that the preamble sequence was not found.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
seq_correlation: float Sequence correlation

Cloning the Group

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

Subgroups

6.3.2.8.1 Preamble<Preamble>

RepCap Settings

```
# Range: Nr1 .. Nr400
rc = driver.prach.modulation.scorrelation.preamble.repcap_preamble_get()
driver.prach.modulation.scorrelation.preamble.repcap_preamble_set(repcap.Preamble.Nr1)
```

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:PRACh:MODulation:SCORrelation:PREamble<Number>
```

class PreambleCls

Preamble commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Preamble, default value after init: Preamble.Nr1

fetch(*preamble=Preamble.Default*) → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:MODulation:SCORrelation:PREamble
↪<Number>
value: float = driver.prach.modulation.scorrelation.preamble.fetch(preamble =␣
↪repcap.Preamble.Default)
```

Returns the sequence correlation for a selected preamble of multi-preamble measurements. It indicates the correlation between the ideal preamble sequence determined from the parameter settings and the measured preamble sequence. A value of 1 corresponds to perfect correlation. A value much smaller than 1 indicates that the preamble sequence was not found.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

param preamble
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Preamble')

return
seq_correlation: float Sequence correlation

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.prach.modulation.scorrelation.preamble.clone()
```

6.3.2.9 StandardDev

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:MODulation:SDEViation
FETCh:LTE:MEASurement<Instance>:PRACh:MODulation:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits. Unit: %
- Evm_Rms_Low: float: float EVM RMS value, low EVM window position. Unit: %
- Evm_Rms_High: float: float EVM RMS value, high EVM window position. Unit: %
- Evm_Peak_Low: float: float EVM peak value, low EVM window position. Unit: %
- Evm_Peak_High: float: float EVM peak value, high EVM window position. Unit: %
- Mag_Error_Rms_Low: float: float Magnitude error RMS value, low EVM window position. Unit: %
- Mag_Error_Rms_High: float: float Magnitude error RMS value, low EVM window position. Unit: %
- Mag_Error_Peak_Low: float: float Magnitude error peak value, low EVM window position. Unit: %
- Mag_Err_Peak_High: float: float Magnitude error peak value, high EVM window position. Unit: %
- Ph_Error_Rms_Low: float: float Phase error RMS value, low EVM window position. Unit: deg
- Ph_Error_Rms_High: float: float Phase error RMS value, high EVM window position. Unit: deg
- Ph_Error_Peak_Low: float: float Phase error peak value, low EVM window position. Unit: deg
- Ph_Error_Peak_High: float: float Phase error peak value, high EVM window position. Unit: deg
- Frequency_Error: float: float Carrier frequency error. Unit: Hz
- Timing_Error: float: float Transmit time error. Unit: Ts (basic LTE time unit)
- Tx_Power: float: float User equipment power. Unit: dBm
- Peak_Power: float: float User equipment peak power. Unit: dBm

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:MODulation:SDEViation
value: ResultData = driver.prach.modulation.standardDev.fetch()
```

Return the current, average and standard deviation single-value 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:LTE:MEASurement<Instance>:PRACh:MODulation:SDEViation
value: ResultData = driver.prach.modulation.standardDev.read()
```

Return the current, average and standard deviation single-value 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.

6.3.3 Pdynamics

class PdynamicsCls

Pdynamics commands group definition. 14 total commands, 5 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.3.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:PDYNamics:AVERage
FETCh:LTE:MEASurement<Instance>:PRACh:PDYNamics:AVERage
CALCulate:LTE:MEASurement<Instance>:PRACh:PDYNamics:AVERage
```

class AverageCls

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

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'

- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for power dynamics measurements exceeding the specified power dynamics limits. Unit: %
- Off_Power_Before: float or bool: float OFF power mean value for subframe before preamble without transient period Unit: dBm
- On_Power_Rms: float or bool: float ON power mean value over preamble Unit: dBm
- On_Power_Peak: float or bool: float ON power peak value within preamble Unit: dBm
- Off_Power_After: float or bool: float OFF power mean value for subframe after preamble without transient period Unit: dBm

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for power dynamics measurements exceeding the specified power dynamics limits. Unit: %
- Off_Power_Before: float: float OFF power mean value for subframe before preamble without transient period Unit: dBm
- On_Power_Rms: float: float ON power mean value over preamble Unit: dBm
- On_Power_Peak: float: float ON power peak value within preamble Unit: dBm
- Off_Power_After: float: float OFF power mean value for subframe after preamble without transient period Unit: dBm

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:PRACh:PDYNamics:AVERage
value: CalculateStruct = driver.prach.pdynamics.average.calculate()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. 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:LTE:MEASurement<Instance>:PRACh:PDYNamics:AVERage
value: ResultData = driver.prach.pdynamics.average.fetch()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. 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:LTE:MEASurement<Instance>:PRACh:PDYNamics:AVERage
value: ResultData = driver.prach.pdynamics.average.read()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. 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.

6.3.3.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:PDYNamics:CURRent
FETCh:LTE:MEASurement<Instance>:PRACh:PDYNamics:CURRent
CALCulate:LTE:MEASurement<Instance>:PRACh:PDYNamics:CURRent
```

class CurrentCls

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

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for power dynamics measurements exceeding the specified power dynamics limits. Unit: %
- Off_Power_Before: float or bool: float OFF power mean value for subframe before preamble without transient period Unit: dBm
- On_Power_Rms: float or bool: float ON power mean value over preamble Unit: dBm
- On_Power_Peak: float or bool: float ON power peak value within preamble Unit: dBm
- Off_Power_After: float or bool: float OFF power mean value for subframe after preamble without transient period Unit: dBm

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for power dynamics measurements exceeding the specified power dynamics limits. Unit: %
- Off_Power_Before: float: float OFF power mean value for subframe before preamble without transient period Unit: dBm
- On_Power_Rms: float: float ON power mean value over preamble Unit: dBm
- On_Power_Peak: float: float ON power peak value within preamble Unit: dBm
- Off_Power_After: float: float OFF power mean value for subframe after preamble without transient period Unit: dBm

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:PRACh:PDYNamics:CURRent
value: CalculateStruct = driver.prach.pdynamics.current.calculate()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. 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:LTE:MEASurement<Instance>:PRACh:PDYNamics:CURRent
value: ResultData = driver.prach.pdynamics.current.fetch()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. 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:LTE:MEASurement<Instance>:PRACh:PDYNamics:CURRent
value: ResultData = driver.prach.pdynamics.current.read()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. 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.

6.3.3.3 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:PDYNamics:MAXimum
FETCh:LTE:MEASurement<Instance>:PRACh:PDYNamics:MAXimum
CALCulate:LTE:MEASurement<Instance>:PRACh:PDYNamics:MAXimum
```

class MaximumCls

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

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for power dynamics measurements exceeding the specified power dynamics limits. Unit: %
- Off_Power_Before: float or bool: float OFF power mean value for subframe before preamble without transient period Unit: dBm
- On_Power_Rms: float or bool: float ON power mean value over preamble Unit: dBm
- On_Power_Peak: float or bool: float ON power peak value within preamble Unit: dBm

- Off_Power_After: float or bool: float OFF power mean value for subframe after preamble without transient period Unit: dBm

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for power dynamics measurements exceeding the specified power dynamics limits. Unit: %
- Off_Power_Before: float: float OFF power mean value for subframe before preamble without transient period Unit: dBm
- On_Power_Rms: float: float ON power mean value over preamble Unit: dBm
- On_Power_Peak: float: float ON power peak value within preamble Unit: dBm
- Off_Power_After: float: float OFF power mean value for subframe after preamble without transient period Unit: dBm

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:PRACh:PDYNamics:MAXimum
value: CalculateStruct = driver.prach.pdynamics.maximum.calculate()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. 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:LTE:MEASurement<Instance>:PRACh:PDYNamics:MAXimum
value: ResultData = driver.prach.pdynamics.maximum.fetch()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. 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:LTE:MEASurement<Instance>:PRACh:PDYNamics:MAXimum
value: ResultData = driver.prach.pdynamics.maximum.read()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. 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.

6.3.3.4 Minimum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:PDYNamics:MINimum
FETCh:LTE:MEASurement<Instance>:PRACh:PDYNamics:MINimum
CALCulate:LTE:MEASurement<Instance>:PRACh:PDYNamics:MINimum
```

class MinimumCls

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

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for power dynamics measurements exceeding the specified power dynamics limits. Unit: %
- Off_Power_Before: float or bool: float OFF power mean value for subframe before preamble without transient period Unit: dBm
- On_Power_Rms: float or bool: float ON power mean value over preamble Unit: dBm
- On_Power_Peak: float or bool: float ON power peak value within preamble Unit: dBm
- Off_Power_After: float or bool: float OFF power mean value for subframe after preamble without transient period Unit: dBm

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for power dynamics measurements exceeding the specified power dynamics limits. Unit: %
- Off_Power_Before: float: float OFF power mean value for subframe before preamble without transient period Unit: dBm
- On_Power_Rms: float: float ON power mean value over preamble Unit: dBm
- On_Power_Peak: float: float ON power peak value within preamble Unit: dBm
- Off_Power_After: float: float OFF power mean value for subframe after preamble without transient period Unit: dBm

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:PRACh:PDYNamics:MINimum
value: CalculateStruct = driver.prach.pdynamics.minimum.calculate()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. 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:LTE:MEASurement<Instance>:PRACh:PDYNamics:MINimum
value: ResultData = driver.prach.pdynamics.minimum.fetch()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. 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:LTE:MEASurement<Instance>:PRACh:PDYNamics:MINimum
value: ResultData = driver.prach.pdynamics.minimum.read()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. 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.

6.3.3.5 StandardDev

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:PDYNamics:SDEViation
FETCh:LTE:MEASurement<Instance>:PRACh:PDYNamics:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for power dynamics measurements exceeding the specified power dynamics limits. Unit: %
- Off_Power_Before: float: float OFF power mean value for subframe before preamble without transient period Unit: dBm
- On_Power_Rms: float: float ON power mean value over preamble Unit: dBm
- On_Power_Peak: float: float ON power peak value within preamble Unit: dBm
- Off_Power_After: float: float OFF power mean value for subframe after preamble without transient period Unit: dBm

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:PDYNamics:SDEViation
value: ResultData = driver.prach.pdynamics.standardDev.fetch()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. 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:LTE:MEASurement<Instance>:PRACh:PDYNamics:SDEViation
value: ResultData = driver.prach.pdynamics.standardDev.read()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. 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.

6.3.4 State

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:PRACh:STATe
```

class StateCls

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

fetch(*timeout: float = None*, *target_main_state: TargetMainState = None*, *target_sync_state: TargetSyncState = None*) → ResourceState

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:STATe
value: enums.ResourceState = driver.prach.state.fetch(timeout = 1.0, target_
↪main_state = enums.TargetMainState.OFF, target_sync_state = enums.
↪TargetSyncState.ADJusted)
```

Queries the main measurement state. Use FETCh:...:STATe:ALL? to query the measurement state including the substates. Use INITiate..., STOP..., ABORt... to change the measurement state.

param timeout
No help available

param target_main_state
No help available

param target_sync_state
No help available

return
meas_status: OFF | RUN | RDY OFF: measurement off, no resources allocated, no results RUN: measurement running, synchronization pending or adjusted, resources active or queued RDY: measurement terminated, valid results can be available

Cloning the Group

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

Subgroups

6.3.4.1 All

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:PRACh:STATe:ALL
```

class AllCls

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

fetch(*timeout: float = None*, *target_main_state: TargetMainState = None*, *target_sync_state: TargetSyncState = None*) → List[ResourceState]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:STATe:ALL
value: List[enums.ResourceState] = driver.prach.state.all.fetch(timeout = 1.0,␣
↪target_main_state = enums.TargetMainState.OFF, target_sync_state = enums.
↪TargetSyncState.ADJusted)
```

Queries the main measurement state and the measurement substates. Both measurement substates are relevant for running measurements only. Use FETCh:...:STATe? to query the main measurement state only. Use INITiate..., STOP..., ABORt... to change the measurement state.

param timeout
No help available

param target_main_state
No help available

param target_sync_state
No help available

return
state: No help available

6.3.5 Trace

class TraceCls

Trace commands group definition. 29 total commands, 7 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.5.1 Evm

class EvmCls

Evm 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.prach.trace.evm.clone()
```

Subgroups

6.3.5.1.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:TRACe:EVM:AVERage
FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:EVM:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:EVM:AVERage
value: List[float] = driver.prach.trace.evm.average.fetch()
```

Return the values of the EVM vs subcarrier traces. Each value is averaged over the samples in one preamble subcarrier. The results of the current, average and maximum traces can be retrieved. See also 'View EVM, Magnitude Error, Phase Error'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
results: float The number of results depends on the preamble format. Format 0 to 3: 839 EVM values, format 4: 139 EVM values Unit: %

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:PRACh:TRACe:EVM:AVERage
value: List[float] = driver.prach.trace.evm.average.read()
```

Return the values of the EVM vs subcarrier traces. Each value is averaged over the samples in one preamble subcarrier. The results of the current, average and maximum traces can be retrieved. See also 'View EVM, Magnitude Error, Phase Error'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
results: float The number of results depends on the preamble format. Format 0 to 3: 839 EVM values, format 4: 139 EVM values Unit: %

6.3.5.1.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:TRACe:EVM:CURRent
FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:EVM:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:EVM:CURRent
value: List[float] = driver.prach.trace.evm.current.fetch()
```

Return the values of the EVM vs subcarrier traces. Each value is averaged over the samples in one preamble subcarrier. The results of the current, average and maximum traces can be retrieved. See also 'View EVM, Magnitude Error, Phase Error'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
results: float The number of results depends on the preamble format. Format 0 to 3: 839 EVM values, format 4: 139 EVM values Unit: %

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:PRACh:TRACe:EVM:CURRent
value: List[float] = driver.prach.trace.evm.current.read()
```

Return the values of the EVM vs subcarrier traces. Each value is averaged over the samples in one preamble subcarrier. The results of the current, average and maximum traces can be retrieved. See also 'View EVM, Magnitude Error, Phase Error'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
results: float The number of results depends on the preamble format. Format 0 to 3: 839 EVM values, format 4: 139 EVM values Unit: %

6.3.5.1.3 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:TRACe:EVM:MAXimum
FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:EVM:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:EVM:MAXimum
value: List[float] = driver.prach.trace.evm.maximum.fetch()
```

Return the values of the EVM vs subcarrier traces. Each value is averaged over the samples in one preamble subcarrier. The results of the current, average and maximum traces can be retrieved. See also 'View EVM, Magnitude Error, Phase Error'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
results: float The number of results depends on the preamble format. Format 0 to 3: 839 EVM values, format 4: 139 EVM values Unit: %

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:PRACh:TRACe:EVM:MAXimum
value: List[float] = driver.prach.trace.evm.maximum.read()
```

Return the values of the EVM vs subcarrier traces. Each value is averaged over the samples in one preamble subcarrier. The results of the current, average and maximum traces can be retrieved. See also 'View EVM, Magnitude Error, Phase Error'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
results: float The number of results depends on the preamble format. Format 0 to 3: 839 EVM values, format 4: 139 EVM values Unit: %

6.3.5.2 EvPreamble

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:TRACe:EVPReamble
FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:EVPReamble
```

class EvPreambleCls

EvPreamble commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:EVPReamble
value: List[float] = driver.prach.trace.evPreamble.fetch()
```

Return the values of the EVM vs preamble traces. See also 'View EVM vs Preamble, Power vs Preamble'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
results: float 32 EVM values, for preamble 1 to 32 (NCAP for not measured preambles)
Unit: %

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:PRACh:TRACe:EVPReamble
value: List[float] = driver.prach.trace.evPreamble.read()
```

Return the values of the EVM vs preamble traces. See also 'View EVM vs Preamble, Power vs Preamble'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
results: float 32 EVM values, for preamble 1 to 32 (NCAP for not measured preambles)
Unit: %

6.3.5.3 Iq

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:IQ
```

class IqCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Iphase: List[float]: float Normalized I amplitude
- Qphase: List[float]: float Normalized Q amplitude

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:IQ
value: FetchStruct = driver.prach.trace.iq.fetch()
```

Returns the results in the I/Q constellation diagram. There is one pair of values per modulation symbol. For preamble format 4, there are 139 symbols. For preamble format 0 to 3, there are 839 symbols. The results are returned in the following order: <Reliability>, {<IPhase>, <QPhase>}symbol 1, . . . , {<IPhase>, <QPhase>}symbol n See also 'View I/Q Constellation'.

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

6.3.5.4 Merror

class MerrorCls

Merror 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.prach.trace.merror.clone()
```

Subgroups

6.3.5.4.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:TRACe:MERRor:AVERage
FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:MERRor:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:MERRor:AVERage
value: List[float] = driver.prach.trace.merror.average.fetch()
```

Return the values of the magnitude error traces. Each value is averaged over the samples in one preamble subcarrier. The results of the current, average and maximum traces can be retrieved. See also 'View EVM, Magnitude Error, Phase Error'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
results: float The number of results depends on the preamble format. Format 0 to 3: 839 EVM values, format 4: 139 EVM values Unit: %

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:PRACh:TRACe:MERRor:AVERage
value: List[float] = driver.prach.trace.merror.average.read()
```

Return the values of the magnitude error traces. Each value is averaged over the samples in one preamble subcarrier. The results of the current, average and maximum traces can be retrieved. See also 'View EVM, Magnitude Error, Phase Error'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
results: float The number of results depends on the preamble format. Format 0 to 3: 839 EVM values, format 4: 139 EVM values Unit: %

6.3.5.4.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:TRACe:MERRor:CURRent
FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:MERRor:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:MERRor:CURRent
value: List[float] = driver.prach.trace.merror.current.fetch()
```

Return the values of the magnitude error traces. Each value is averaged over the samples in one preamble subcarrier. The results of the current, average and maximum traces can be retrieved. See also 'View EVM, Magnitude Error, Phase Error'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
results: float The number of results depends on the preamble format. Format 0 to 3: 839 EVM values, format 4: 139 EVM values Unit: %

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:PRACh:TRACe:MERRor:CURRent
value: List[float] = driver.prach.trace.merror.current.read()
```

Return the values of the magnitude error traces. Each value is averaged over the samples in one preamble subcarrier. The results of the current, average and maximum traces can be retrieved. See also 'View EVM, Magnitude Error, Phase Error'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
results: float The number of results depends on the preamble format. Format 0 to 3: 839 EVM values, format 4: 139 EVM values Unit: %

6.3.5.4.3 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:TRACe:MERRor:MAXimum
FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:MERRor:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:MERRor:MAXimum
value: List[float] = driver.prach.trace.merror.maximum.fetch()
```

Return the values of the magnitude error traces. Each value is averaged over the samples in one preamble subcarrier. The results of the current, average and maximum traces can be retrieved. See also 'View EVM, Magnitude Error, Phase Error'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
results: float The number of results depends on the preamble format. Format 0 to 3: 839 EVM values, format 4: 139 EVM values Unit: %

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:PRACh:TRACe:MERRor:MAXimum
value: List[float] = driver.prach.trace.merror.maximum.read()
```

Return the values of the magnitude error traces. Each value is averaged over the samples in one preamble subcarrier. The results of the current, average and maximum traces can be retrieved. See also 'View EVM, Magnitude Error, Phase Error'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
results: float The number of results depends on the preamble format. Format 0 to 3: 839 EVM values, format 4: 139 EVM values Unit: %

6.3.5.5 Pdynamics

class PdynamicsCls

Pdynamics 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.prach.trace.pdynamics.clone()
```

Subgroups

6.3.5.5.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:TRACe:PDYNamics:AVERage
FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:PDYNamics:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:PDYNamics:AVERage
value: List[float] = driver.prach.trace.pdynamics.average.fetch()
```

Return the values of the power dynamics traces. Each value is sampled with 48 Ts, corresponding to 1.5625 µs. The results of the current, average and maximum traces can be retrieved. See also 'View Power Dynamics'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
power: float 2048 power values, from -1100 µs to +2098.4375 µs relative to the start of the preamble. The values have a spacing of 1.5625 µs. The 705th value is at the start of the preamble (0 µs) . Unit: dBm

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:PRACh:TRACe:PDYNamics:AVERage
value: List[float] = driver.prach.trace.pdynamics.average.read()
```

Return the values of the power dynamics traces. Each value is sampled with 48 Ts, corresponding to 1.5625 µs. The results of the current, average and maximum traces can be retrieved. See also 'View Power Dynamics'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
power: float 2048 power values, from -1100 µs to +2098.4375 µs relative to the start of the preamble. The values have a spacing of 1.5625 µs. The 705th value is at the start of the preamble (0 µs) . Unit: dBm

6.3.5.5.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:TRACe:PDYNamics:CURRent
FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:PDYNamics:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:PDYNamics:CURRent
value: List[float] = driver.prach.trace.pdynamics.current.fetch()
```

Return the values of the power dynamics traces. Each value is sampled with 48 Ts, corresponding to 1.5625 µs. The results of the current, average and maximum traces can be retrieved. See also 'View Power Dynamics'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
power: float 2048 power values, from -1100 µs to +2098.4375 µs relative to the start of the preamble. The values have a spacing of 1.5625 µs. The 705th value is at the start of the preamble (0 µs) . Unit: dBm

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:PRACh:TRACe:PDYNamics:CURRent
value: List[float] = driver.prach.trace.pdynamics.current.read()
```

Return the values of the power dynamics traces. Each value is sampled with 48 Ts, corresponding to 1.5625 µs. The results of the current, average and maximum traces can be retrieved. See also 'View Power Dynamics'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
power: float 2048 power values, from -1100 µs to +2098.4375 µs relative to the start of the preamble. The values have a spacing of 1.5625 µs. The 705th value is at the start of the preamble (0 µs) . Unit: dBm

6.3.5.5.3 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:TRACe:PDYNamics:MAXimum
FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:PDYNamics:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:PDYNamics:MAXimum
value: List[float] = driver.prach.trace.pdynamics.maximum.fetch()
```

Return the values of the power dynamics traces. Each value is sampled with 48 Ts, corresponding to 1.5625 µs. The results of the current, average and maximum traces can be retrieved. See also 'View Power Dynamics'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
power: float 2048 power values, from -1100 µs to +2098.4375 µs relative to the start of the preamble. The values have a spacing of 1.5625 µs. The 705th value is at the start of the preamble (0 µs) . Unit: dBm

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:PRACh:TRACe:PDYNamics:MAXimum
value: List[float] = driver.prach.trace.pdynamics.maximum.read()
```

Return the values of the power dynamics traces. Each value is sampled with 48 Ts, corresponding to 1.5625 µs. The results of the current, average and maximum traces can be retrieved. See also 'View Power Dynamics'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
power: float 2048 power values, from -1100 µs to +2098.4375 µs relative to the start of the preamble. The values have a spacing of 1.5625 µs. The 705th value is at the start of the preamble (0 µs) . Unit: dBm

6.3.5.6 Perror

class PerrorCls

Perror 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.prach.trace.perror.clone()
```

Subgroups

6.3.5.6.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:TRACe:PERRor:AVERage
FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:PERRor:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:PERRor:AVERage
value: List[float] = driver.prach.trace.perror.average.fetch()
```

Return the values of the phase error traces. Each value is averaged over the samples in one preamble subcarrier. The results of the current, average and maximum traces can be retrieved. See also 'View EVM, Magnitude Error, Phase Error'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
results: float The number of results depends on the preamble format. Format 0 to 3: 839 EVM values, format 4: 139 EVM values Unit: deg

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:PRACh:TRACe:PERRor:AVERage
value: List[float] = driver.prach.trace.perror.average.read()
```

Return the values of the phase error traces. Each value is averaged over the samples in one preamble subcarrier. The results of the current, average and maximum traces can be retrieved. See also 'View EVM, Magnitude Error, Phase Error'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
results: float The number of results depends on the preamble format. Format 0 to 3: 839 EVM values, format 4: 139 EVM values Unit: deg

6.3.5.6.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:TRACe:PERRor:CURRent
FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:PERRor:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:PERRor:CURRent
value: List[float] = driver.prach.trace.perror.current.fetch()
```

Return the values of the phase error traces. Each value is averaged over the samples in one preamble subcarrier. The results of the current, average and maximum traces can be retrieved. See also 'View EVM, Magnitude Error, Phase Error'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
results: float The number of results depends on the preamble format. Format 0 to 3: 839 EVM values, format 4: 139 EVM values Unit: deg

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:PRACh:TRACe:PERRor:CURRent
value: List[float] = driver.prach.trace.perror.current.read()
```

Return the values of the phase error traces. Each value is averaged over the samples in one preamble subcarrier. The results of the current, average and maximum traces can be retrieved. See also 'View EVM, Magnitude Error, Phase Error'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
results: float The number of results depends on the preamble format. Format 0 to 3: 839 EVM values, format 4: 139 EVM values Unit: deg

6.3.5.6.3 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:TRACe:PERRor:MAXimum
FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:PERRor:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:PERRor:MAXimum
value: List[float] = driver.prach.trace.perror.maximum.fetch()
```

Return the values of the phase error traces. Each value is averaged over the samples in one preamble subcarrier. The results of the current, average and maximum traces can be retrieved. See also 'View EVM, Magnitude Error, Phase Error'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
results: float The number of results depends on the preamble format. Format 0 to 3: 839 EVM values, format 4: 139 EVM values Unit: deg

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:PRACh:TRACe:PERRor:MAXimum
value: List[float] = driver.prach.trace.perror.maximum.read()
```

Return the values of the phase error traces. Each value is averaged over the samples in one preamble subcarrier. The results of the current, average and maximum traces can be retrieved. See also 'View EVM, Magnitude Error, Phase Error'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
results: float The number of results depends on the preamble format. Format 0 to 3: 839 EVM values, format 4: 139 EVM values Unit: deg

6.3.5.7 PvPreamble

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:TRACe:PVPReamble
FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:PVPReamble
```

class PvPreambleCls

PvPreamble commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:PVPReamble
value: List[float] = driver.prach.trace.pvPreamble.fetch()
```

Return the values of the power vs preamble traces. See also 'View EVM vs Preamble, Power vs Preamble'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
results: float 32 power values, for preamble 1 to 32 (NCAP for not measured preambles) Unit: dBm

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:PRACh:TRACe:PVPReamble
value: List[float] = driver.prach.trace.pvPreamble.read()
```

Return the values of the power vs preamble traces. See also 'View EVM vs Preamble, Power vs Preamble'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
results: float 32 power values, for preamble 1 to 32 (NCAP for not measured preambles)
Unit: dBm

6.4 Route

SCPI Command :

```
ROUTe:LTE:MEASurement<Instance>
```

class RouteCls

Route commands group definition. 5 total commands, 1 Subgroups, 1 group commands

class ValueStruct

Structure for reading output parameters. Fields:

- Scenario: enums.Scenario: SALone | CSPath | MAPRotocol SALone: Standalone (non-signaling) CSPath: Combined signal path MAPRotocol: Measure at protocol test
- Controller: str: string Controlling application for scenario CSPath or MAPRotocol
- Rx_Connector: enums.RxConnector: RF connector for the input path
- Rf_Converter: enums.RxConverter: RX module for the input path

get_value() → ValueStruct

```
# SCPI: ROUTe:LTE:MEASurement<Instance>
value: ValueStruct = driver.route.get_value()
```

Returns the configured routing settings. For possible connector and converter values, see 'Values for RF path selection'.

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.clone()
```

Subgroups

6.4.1 Scenario

SCPI Command :

```
ROUTe:LTE:MEASurement<Instance>:SCENario
```

class ScenarioCls

Scenario commands group definition. 4 total commands, 3 Subgroups, 1 group commands

get_value() → Scenario

```
# SCPI: ROUTe:LTE:MEASurement<Instance>:SCENario
value: enums.Scenario = driver.route.scenario.get_value()
```

Returns the active scenario.

return
scenario: SALone | CSPath | MAPRotocol SALone: Standalone (non-signaling) CSPath: Combined signal path MAPRotocol: Measure at protocol test

Cloning the Group

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

Subgroups

6.4.1.1 CombinedSignalPath

SCPI Command :

```
ROUTe:LTE:MEASurement<Instance>:SCENario:CSPath
```

class CombinedSignalPathCls

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

class GetStruct

Response structure. Fields:

- Master: str: string Controlling application Example: ‘LTE Sig1’ or ‘LTE Sig2’
- Carrier: str: string Uplink carrier or set of uplink carriers configured in the controlling application Examples: ‘PCC’, ‘SCC2’, ‘Set A’, ‘Set B’ If a set is selected, a query returns the carrier of the set that is used by the measurement to select the RF path.
- Set_Py: str: string Measured set of uplink carriers Example: ‘Set A’, ‘Set B’ Only returned for intra-band contiguous UL CA.

get() → GetStruct

```
# SCPI: ROUTe:LTE:MEASurement<Instance>:SCENario:CSPath
value: GetStruct = driver.route.scenario.combinedSignalPath.get()
```

Activates the combined signal path scenario and selects the controlling application and primary carrier. The selected application controls most signal routing settings, analyzer settings and some measurement control settings while the combined signal path scenario is active. The command usage depends on the carrier aggregation mode of the measured signal: no UL carrier aggregation, non-contiguous UL carrier aggregation or intraband contiguous UL carrier aggregation. The following table provides an overview.

Table Header: CA type / Setting command / Query returns

- No UL CA / ROUT:LTE:MEAS:SCEN:CSP <Controller> <Carrier> can be skipped and equals ‘PCC’. / <Controller>, ‘PCC’

- Non-contiguous UL CA / ROUT:LTE:MEAS:SCEN:CSP <Controller>, <Carrier> <Carrier>: measured carrier ('PCC', 'SCC2', ...) / <Controller>, <Carrier>
- Intraband contiguous UL CA / ROUT:LTE:MEAS:SCEN:CSP <Controller>, <Carrier> <Carrier>: set of carriers ('Set A', 'Set B', ...) / <Controller>, <Carrier>, <Set> <Carrier>: carrier selecting RF path ('PCC', 'SCC2', ...) <Set>: measured set of carriers ('Set A', 'Set B', ...)

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

set(*master: str*, *carrier: str = None*) → None

```
# SCPI: ROUTe:LTE:MEASurement<Instance>:SCENario:CSPath
driver.route.scenario.combinedSignalPath.set(master = 'abc', carrier = 'abc')
```

Activates the combined signal path scenario and selects the controlling application and primary carrier. The selected application controls most signal routing settings, analyzer settings and some measurement control settings while the combined signal path scenario is active. The command usage depends on the carrier aggregation mode of the measured signal: no UL carrier aggregation, non-contiguous UL carrier aggregation or intraband contiguous UL carrier aggregation. The following table provides an overview.

Table Header: CA type / Setting command / Query returns

- No UL CA / ROUT:LTE:MEAS:SCEN:CSP <Controller> <Carrier> can be skipped and equals 'PCC'. / <Controller>, 'PCC'
- Non-contiguous UL CA / ROUT:LTE:MEAS:SCEN:CSP <Controller>, <Carrier> <Carrier>: measured carrier ('PCC', 'SCC2', ...) / <Controller>, <Carrier>
- Intraband contiguous UL CA / ROUT:LTE:MEAS:SCEN:CSP <Controller>, <Carrier> <Carrier>: set of carriers ('Set A', 'Set B', ...) / <Controller>, <Carrier>, <Set> <Carrier>: carrier selecting RF path ('PCC', 'SCC2', ...) <Set>: measured set of carriers ('Set A', 'Set B', ...)

param master
No help available

param carrier
string Uplink carrier or set of uplink carriers configured in the controlling application Examples: 'PCC', 'SCC2', 'Set A', 'Set B' If a set is selected, a query returns the carrier of the set that is used by the measurement to select the RF path.

6.4.1.2 MaProtocol

SCPI Command :

```
ROUTe:LTE:MEASurement<Instance>:SCENario:MAPRotocol
```

class MaProtocolCls

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

set(*controler: str = None*) → None

```
# SCPI: ROUTe:LTE:MEASurement<Instance>:SCENario:MAPRotocol
driver.route.scenario.maProtocol.set(controler = 'abc')
```

Activates the Measure@ProtocolTest scenario and optionally selects the controlling protocol test application. The signal routing and analyzer settings are ignored by the measurement application. Configure the corresponding settings within the protocol test application used in parallel.

param controler
string Protocol test application Example: 'Protocol Test1'

6.4.1.3 Salone

SCPI Command :

```
ROUTe:LTE: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.RxConnector: RF connector for the input path
- Rf_Converter: enums.RxConverter: RX module for the input path

`get()` → SaloneStruct

```
# SCPI: ROUTe:LTE:MEASurement<Instance>:SCENario:SALone
value: SaloneStruct = driver.route.scenario.salone.get()
```

Activates the standalone scenario and selects the RF input path for the measured RF signal. For possible connector and converter values, see 'Values for RF path selection'.

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

`set(`*rx_connector: RxConnector*, *rf_converter: RxConverter*`)` → None

```
# SCPI: ROUTe:LTE:MEASurement<Instance>:SCENario:SALone
driver.route.scenario.salone.set(rx_connector = enums.RxConnector.I11I, rf_
↪converter = enums.RxConverter.IRX1)
```

Activates the standalone scenario and selects the RF input path for the measured RF signal. For possible connector and converter values, see 'Values for RF path selection'.

param rx_connector
RF connector for the input path

param rf_converter
RX module for the input path

6.5 Sense

class SenseCls

Sense 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.sense.clone()
```

Subgroups

6.5.1 CarrierAggregation

SCPI Command :

```
SENSe:LTE:MEASurement<Instance>:CAGGregation:FSHWare
```

class CarrierAggregationCls

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

get_fshware() → bool

```
# SCPI: SENSe:LTE:MEASurement<Instance>:CAGGregation:FSHWare
value: bool = driver.sense.carrierAggregation.get_fshware()
```

This command is only relevant for combined signal path measurements in multi-CMW setups. It queries whether the measurement instance and the carrier to be measured are in the same CMW. If they are in different CMWs, the measurement fails. To correct the problem, use another measurement instance or select another carrier, so that both are in the same CMW.

return

value: OFF | ON OFF: Different CMWs - error ON: Same CMW - ok

6.5.2 MultiEval

class MultiEvalCls

MultiEval 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.sense.multiEval.clone()
```

Subgroups

6.5.2.1 Spectrum

class SpectrumCls

Spectrum 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.sense.multiEval.spectrum.clone()
```

Subgroups

6.5.2.1.1 SeMask

class SeMaskCls

SeMask 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.sense.multiEval.spectrum.seMask.clone()
```

Subgroups

6.5.2.1.1.1 Rbw

SCPI Command :

```
SENSe:LTE:MEASurement<Instance>:MEValuation:SPECtrum:SEMask:RBW:USED
```

class RbwCls

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

class UsedStruct

Structure for reading output parameters. Fields:

- Trace_1: int: decimal RBW for trace 1 (smallest RBW) Unit: kHz
- Trace_2: int: decimal RBW for trace 2 (intermediate RBW) Unit: kHz
- Trace_3: int: decimal RBW for trace 3 (largest RBW) Unit: kHz

get_used() → UsedStruct

```
# SCPI: SENSe:LTE:MEASurement<Instance>:MEValuation:SPECtrum:SEMask:RBW:USED
value: UsedStruct = driver.sense.multiEval.spectrum.seMask.rbw.get_used()
```

Queries the resolution bandwidths (RBW) allowed for spectrum emission measurements. The RBWs depend on the channel bandwidth and on the network signaled value.

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

6.6 Srs

SCPI Commands :

```
INITiate:LTE:MEASurement<Instance>:SRS
STOP:LTE:MEASurement<Instance>:SRS
ABORt:LTE:MEASurement<Instance>:SRS
```

class SrsCls

Srs commands group definition. 25 total commands, 3 Subgroups, 3 group commands

abort(*opc_timeout_ms: int = -1*) → None

```
# SCPI: ABORt:LTE:MEASurement<Instance>:SRS
driver.srs.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
↪' 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:LTE:MEASurement<Instance>:SRS
driver.srs.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
```

(continues on next page)

(continued from previous page)

```
↪'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() → None

```
# SCPI: STOP:LTE:MEASurement<Instance>:SRS
driver.srs.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.

stop_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: STOP:LTE:MEASurement<Instance>:SRS
driver.srs.stop_with_opc()


    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.

Same as stop, but waits for the operation to complete before continuing further. Use the RsCmwLteMeas.utilities.opc_timeout_set() to set the timeout value.

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.srs.clone()
```

Subgroups

6.6.1 Pdynamics

class PdynamicsCls

Pdynamics commands group definition. 14 total commands, 5 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.6.1.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:SRS:PDYNamics:AVERage
FETCh:LTE:MEASurement<Instance>:SRS:PDYNamics:AVERage
CALCulate:LTE:MEASurement<Instance>:SRS:PDYNamics:AVERage
```

class AverageCls

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

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for power dynamics measurements exceeding the specified power dynamics limits. Unit: %
- Off_Power_Before: float or bool: float OFF power mean value for time period before SRS symbol Unit: dBm
- On_Power_Rms_1: float or bool: float ON power mean value over the first SRS symbol Unit: dBm
- On_Power_Peak_1: float or bool: float ON power peak value for the first SRS symbol Unit: dBm
- On_Power_Rms_2: float or bool: float ON power mean value over the second SRS symbol (NCAP returned for FDD) Unit: dBm
- On_Power_Peak_2: float or bool: float ON power peak value for the second SRS symbol (NCAP returned for FDD) Unit: dBm

- Off_Power_After: float or bool: float OFF power mean value for subframe after SRS symbol Unit: dBm

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for power dynamics measurements exceeding the specified power dynamics limits. Unit: %
- Off_Power_Before: float: float OFF power mean value for time period before SRS symbol Unit: dBm
- On_Power_Rms_1: float: float ON power mean value over the first SRS symbol Unit: dBm
- On_Power_Peak_1: float: float ON power peak value for the first SRS symbol Unit: dBm
- On_Power_Rms_2: float: float ON power mean value over the second SRS symbol (NCAP returned for FDD) Unit: dBm
- On_Power_Peak_2: float: float ON power peak value for the second SRS symbol (NCAP returned for FDD) Unit: dBm
- Off_Power_After: float: float OFF power mean value for subframe after SRS symbol Unit: dBm

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:SRS:PDYNamics:AVERage
value: CalculateStruct = driver.srs.pdynamics.average.calculate()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. 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:LTE:MEASurement<Instance>:SRS:PDYNamics:AVERage
value: ResultData = driver.srs.pdynamics.average.fetch()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. 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:LTE:MEASurement<Instance>:SRS:PDYNamics:AVERage
value: ResultData = driver.srs.pdynamics.average.read()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. 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.

6.6.1.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:SRS:PDYNamics:CURRent
FETCh:LTE:MEASurement<Instance>:SRS:PDYNamics:CURRent
CALCulate:LTE:MEASurement<Instance>:SRS:PDYNamics:CURRent
```

`class CurrentCls`

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

`class CalculateStruct`

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for power dynamics measurements exceeding the specified power dynamics limits. Unit: %
- Off_Power_Before: float or bool: float OFF power mean value for time period before SRS symbol Unit: dBm
- On_Power_Rms_1: float or bool: float ON power mean value over the first SRS symbol Unit: dBm
- On_Power_Peak_1: float or bool: float ON power peak value for the first SRS symbol Unit: dBm
- On_Power_Rms_2: float or bool: float ON power mean value over the second SRS symbol (NCAP returned for FDD) Unit: dBm
- On_Power_Peak_2: float or bool: float ON power peak value for the second SRS symbol (NCAP returned for FDD) Unit: dBm
- Off_Power_After: float or bool: float OFF power mean value for subframe after SRS symbol Unit: dBm

`class ResultData`

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for power dynamics measurements exceeding the specified power dynamics limits. Unit: %
- Off_Power_Before: float: float OFF power mean value for time period before SRS symbol Unit: dBm
- On_Power_Rms_1: float: float ON power mean value over the first SRS symbol Unit: dBm
- On_Power_Peak_1: float: float ON power peak value for the first SRS symbol Unit: dBm
- On_Power_Rms_2: float: float ON power mean value over the second SRS symbol (NCAP returned for FDD) Unit: dBm
- On_Power_Peak_2: float: float ON power peak value for the second SRS symbol (NCAP returned for FDD) Unit: dBm
- Off_Power_After: float: float OFF power mean value for subframe after SRS symbol Unit: dBm

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:SRS:PDYNamics:CURRent
value: CalculateStruct = driver.srs.pdynamics.current.calculate()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. 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:LTE:MEASurement<Instance>:SRS:PDYNamics:CURRent
value: ResultData = driver.srs.pdynamics.current.fetch()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. 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:LTE:MEASurement<Instance>:SRS:PDYNamics:CURRent
value: ResultData = driver.srs.pdynamics.current.read()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. 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.

6.6.1.3 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:SRS:PDYNamics:MAXimum
FETCh:LTE:MEASurement<Instance>:SRS:PDYNamics:MAXimum
CALCulate:LTE:MEASurement<Instance>:SRS:PDYNamics:MAXimum
```

class MaximumCls

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

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for power dynamics measurements exceeding the specified power dynamics limits. Unit: %

- Off_Power_Before: float or bool: float OFF power mean value for time period before SRS symbol Unit: dBm
- On_Power_Rms_1: float or bool: float ON power mean value over the first SRS symbol Unit: dBm
- On_Power_Peak_1: float or bool: float ON power peak value for the first SRS symbol Unit: dBm
- On_Power_Rms_2: float or bool: float ON power mean value over the second SRS symbol (NCAP returned for FDD) Unit: dBm
- On_Power_Peak_2: float or bool: float ON power peak value for the second SRS symbol (NCAP returned for FDD) Unit: dBm
- Off_Power_After: float or bool: float OFF power mean value for subframe after SRS symbol Unit: dBm

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for power dynamics measurements exceeding the specified power dynamics limits. Unit: %
- Off_Power_Before: float: float OFF power mean value for time period before SRS symbol Unit: dBm
- On_Power_Rms_1: float: float ON power mean value over the first SRS symbol Unit: dBm
- On_Power_Peak_1: float: float ON power peak value for the first SRS symbol Unit: dBm
- On_Power_Rms_2: float: float ON power mean value over the second SRS symbol (NCAP returned for FDD) Unit: dBm
- On_Power_Peak_2: float: float ON power peak value for the second SRS symbol (NCAP returned for FDD) Unit: dBm
- Off_Power_After: float: float OFF power mean value for subframe after SRS symbol Unit: dBm

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:SRS:PDYNamics:MAXimum
value: CalculateStruct = driver.srs.pdynamics.maximum.calculate()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. 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:LTE:MEASurement<Instance>:SRS:PDYNamics:MAXimum
value: ResultData = driver.srs.pdynamics.maximum.fetch()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. 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:LTE:MEASurement<Instance>:SRS:PDYNamics:MAXimum
value: ResultData = driver.srs.pdynamics.maximum.read()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. 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.

6.6.1.4 Minimum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:SRS:PDYNamics:MINimum
FETCh:LTE:MEASurement<Instance>:SRS:PDYNamics:MINimum
CALCulate:LTE:MEASurement<Instance>:SRS:PDYNamics:MINimum
```

class MinimumCls

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

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for power dynamics measurements exceeding the specified power dynamics limits. Unit: %
- Off_Power_Before: float or bool: float OFF power mean value for time period before SRS symbol Unit: dBm
- On_Power_Rms_1: float or bool: float ON power mean value over the first SRS symbol Unit: dBm
- On_Power_Peak_1: float or bool: float ON power peak value for the first SRS symbol Unit: dBm
- On_Power_Rms_2: float or bool: float ON power mean value over the second SRS symbol (NCAP returned for FDD) Unit: dBm
- On_Power_Peak_2: float or bool: float ON power peak value for the second SRS symbol (NCAP returned for FDD) Unit: dBm
- Off_Power_After: float or bool: float OFF power mean value for subframe after SRS symbol Unit: dBm

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for power dynamics measurements exceeding the specified power dynamics limits. Unit: %
- Off_Power_Before: float: float OFF power mean value for time period before SRS symbol Unit: dBm
- On_Power_Rms_1: float: float ON power mean value over the first SRS symbol Unit: dBm

- On_Power_Peak_1: float: float ON power peak value for the first SRS symbol Unit: dBm
- On_Power_Rms_2: float: float ON power mean value over the second SRS symbol (NCAP returned for FDD) Unit: dBm
- On_Power_Peak_2: float: float ON power peak value for the second SRS symbol (NCAP returned for FDD) Unit: dBm
- Off_Power_After: float: float OFF power mean value for subframe after SRS symbol Unit: dBm

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:SRS:PDYNamics:MINimum
value: CalculateStruct = driver.srs.pdynamics.minimum.calculate()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. 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:LTE:MEASurement<Instance>:SRS:PDYNamics:MINimum
value: ResultData = driver.srs.pdynamics.minimum.fetch()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. 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:LTE:MEASurement<Instance>:SRS:PDYNamics:MINimum
value: ResultData = driver.srs.pdynamics.minimum.read()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. 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.

6.6.1.5 StandardDev

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:SRS:PDYNamics:SDEViation
FETCh:LTE:MEASurement<Instance>:SRS:PDYNamics:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability indicator'
- Out_Of_Tolerance: int: decimal Out of tolerance result, i.e. percentage of measurement intervals of the statistic count for power dynamics measurements exceeding the specified power dynamics limits. Unit: %
- Off_Power_Before: float: float OFF power mean value for time period before SRS symbol Unit: dBm
- On_Power_Rms_1: float: float ON power mean value over the first SRS symbol Unit: dBm
- On_Power_Peak_1: float: float ON power peak value for the first SRS symbol Unit: dBm
- On_Power_Rms_2: float: float ON power mean value over the second SRS symbol (NCAP returned for FDD) Unit: dBm
- On_Power_Peak_2: float: float ON power peak value for the second SRS symbol (NCAP returned for FDD) Unit: dBm
- Off_Power_After: float: float OFF power mean value for subframe after SRS symbol Unit: dBm

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:SRS:PDYNamics:SDEViation
value: ResultData = driver.srs.pdynamics.standardDev.fetch()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. 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:LTE:MEASurement<Instance>:SRS:PDYNamics:SDEViation
value: ResultData = driver.srs.pdynamics.standardDev.read()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. 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.

6.6.2 State

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:SRS:STATe
```

class StateCls

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

fetch(*timeout: float = None*, *target_main_state: TargetMainState = None*, *target_sync_state: TargetSyncState = None*) → ResourceState

```
# SCPI: FETCh:LTE:MEASurement<Instance>:SRS:STATe
value: enums.ResourceState = driver.srs.state.fetch(timeout = 1.0, target_main_
↪state = enums.TargetMainState.OFF, target_sync_state = enums.TargetSyncState.
↪ADJusted)
```

Queries the main measurement state. Use FETCh:...:STATe:ALL? to query the measurement state including the substates. Use INITiate..., STOP..., ABORt... to change the measurement state.

param timeout
No help available

param target_main_state
No help available

param target_sync_state
No help available

return
meas_status: OFF | RUN | RDY OFF: measurement off, no resources allocated, no results RUN: measurement running, synchronization pending or adjusted, resources active or queued RDY: measurement terminated, valid results can be available

Cloning the Group

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

Subgroups

6.6.2.1 All

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:SRS:STATe:ALL
```

class AllCls

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

fetch(*timeout: float = None*, *target_main_state: TargetMainState = None*, *target_sync_state: TargetSyncState = None*) → List[ResourceState]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:SRS:STATe:ALL
value: List[enums.ResourceState] = driver.srs.state.all.fetch(timeout = 1.0,␣
↪target_main_state = enums.TargetMainState.OFF, target_sync_state = enums.
↪TargetSyncState.ADJusted)
```

Queries the main measurement state and the measurement substates. Both measurement substates are relevant for running measurements only. Use FETCh:...:STATe? to query the main measurement state only. Use INITiate..., STOP..., ABORt... to change the measurement state.

param timeout
No help available

param target_main_state
No help available

param target_sync_state
No help available

return
state: No help available

6.6.3 Trace

class TraceCls

Trace 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.srs.trace.clone()
```

Subgroups

6.6.3.1 Pdynamics

class PdynamicsCls

Pdynamics 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.srs.trace.pdynamics.clone()
```

Subgroups

6.6.3.1.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:SRS:TRACe:PDYNamics:AVERage
FETCh:LTE:MEASurement<Instance>:SRS:TRACe:PDYNamics:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:SRS:TRACe:PDYNamics:AVERage
value: List[float] = driver.srs.trace.pdynamics.average.fetch()
```

Return the values of the power dynamics traces. Each value is sampled with 48 Ts, corresponding to 1.5625 µs. The results of the current, average and maximum traces can be retrieved. Note that the GUI shows only the beginning of the trace returned via remote command. The last 800 µs cannot be displayed at the GUI. See also 'Measurement results'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
power: float 2048 power values, from -1100 µs to +2098.4375 µs relative to the start of the SRS symbol. The values have a spacing of 1.5625 µs. The 705th value is at the start of the SRS symbol (0 µs) . Unit: dBm

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:SRS:TRACe:PDYNamics:AVERage
value: List[float] = driver.srs.trace.pdynamics.average.read()
```

Return the values of the power dynamics traces. Each value is sampled with 48 Ts, corresponding to 1.5625 µs. The results of the current, average and maximum traces can be retrieved. Note that the GUI shows only the beginning of the trace returned via remote command. The last 800 µs cannot be displayed at the GUI. See also 'Measurement results'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
power: float 2048 power values, from -1100 µs to +2098.4375 µs relative to the start of the SRS symbol. The values have a spacing of 1.5625 µs. The 705th value is at the start of the SRS symbol (0 µs) . Unit: dBm

6.6.3.1.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:SRS:TRACe:PDYNamics:CURRent
FETCh:LTE:MEASurement<Instance>:SRS:TRACe:PDYNamics:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:SRS:TRACe:PDYNamics:CURRent
value: List[float] = driver.srs.trace.pdynamics.current.fetch()
```

Return the values of the power dynamics traces. Each value is sampled with 48 Ts, corresponding to 1.5625 µs. The results of the current, average and maximum traces can be retrieved. Note that the GUI shows only the beginning of the trace returned via remote command. The last 800 µs cannot be displayed at the GUI. See also 'Measurement results'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
power: float 2048 power values, from -1100 µs to +2098.4375 µs relative to the start

of the SRS symbol. The values have a spacing of 1.5625 µs. The 705th value is at the start of the SRS symbol (0 µs) . Unit: dBm

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:SRS:TRACe:PDYNamics:CURRent
value: List[float] = driver.srs.trace.pdynamics.current.read()
```

Return the values of the power dynamics traces. Each value is sampled with 48 Ts, corresponding to 1.5625 µs. The results of the current, average and maximum traces can be retrieved. Note that the GUI shows only the beginning of the trace returned via remote command. The last 800 µs cannot be displayed at the GUI. See also 'Measurement results'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
power: float 2048 power values, from -1100 µs to +2098.4375 µs relative to the start of the SRS symbol. The values have a spacing of 1.5625 µs. The 705th value is at the start of the SRS symbol (0 µs) . Unit: dBm

6.6.3.1.3 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:SRS:TRACe:PDYNamics:MAXimum
FETCh:LTE:MEASurement<Instance>:SRS:TRACe:PDYNamics:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:SRS:TRACe:PDYNamics:MAXimum
value: List[float] = driver.srs.trace.pdynamics.maximum.fetch()
```

Return the values of the power dynamics traces. Each value is sampled with 48 Ts, corresponding to 1.5625 µs. The results of the current, average and maximum traces can be retrieved. Note that the GUI shows only the beginning of the trace returned via remote command. The last 800 µs cannot be displayed at the GUI. See also 'Measurement results'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
power: float 2048 power values, from -1100 µs to +2098.4375 µs relative to the start of the SRS symbol. The values have a spacing of 1.5625 µs. The 705th value is at the start of the SRS symbol (0 µs) . Unit: dBm

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:SRS:TRACe:PDYNamics:MAXimum
value: List[float] = driver.srs.trace.pdynamics.maximum.read()
```

Return the values of the power dynamics traces. Each value is sampled with 48 Ts, corresponding to 1.5625 µs. The results of the current, average and maximum traces can be retrieved. Note that the GUI shows only the beginning of the trace returned via remote command. The last 800 µs cannot be displayed at the GUI. See also 'Measurement results'.

Use RsCmwLteMeas.reliability.last_value to read the updated reliability indicator.

return
power: float 2048 power values, from -1100 µs to +2098.4375 µs relative to the start of the SRS symbol. The values have a spacing of 1.5625 µs. The 705th value is at the start of the SRS symbol (0 µs) . Unit: dBm

6.7 Trigger

class TriggerCls
Trigger commands group definition. 23 total commands, 3 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.7.1 MultiEval

SCPI Commands :

```
TRIGger:LTE:MEASurement<Instance>:MEValuation:SOURce
TRIGger:LTE:MEASurement<Instance>:MEValuation:THReshold
TRIGger:LTE:MEASurement<Instance>:MEValuation:SLOPe
TRIGger:LTE:MEASurement<Instance>:MEValuation:DELay
TRIGger:LTE:MEASurement<Instance>:MEValuation:TOUT
TRIGger:LTE:MEASurement<Instance>:MEValuation:MGAP
TRIGger:LTE:MEASurement<Instance>:MEValuation:SMODe
TRIGger:LTE:MEASurement<Instance>:MEValuation:AMODe
```

class MultiEvalCls
MultiEval commands group definition. 11 total commands, 2 Subgroups, 8 group commands

get_amode() → MevAcquisitionMode

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:MEValuation:AMODe
value: enums.MevAcquisitionMode = driver.trigger.multiEval.get_amode()
```

Selects whether the measurement synchronizes to a slot boundary or to a subframe boundary. The parameter is relevant for 'Free Run (Fast Sync) ' and for list mode measurements with 'Synchronization Mode' = 'Enhanced'.

return
acquisition_mode: SLOT | SUBFrame

get_delay() → float

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:MEValuation:DELay
value: float = driver.trigger.multiEval.get_delay()
```

Defines a time delaying the start of the measurement relative to the trigger event. This setting has no influence on free run measurements.

return
delay: numeric Range: -250E-6 s to 250E-6 s, Unit: s

get_mgap() → int

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:MEValuation:MGAP
value: int = driver.trigger.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: integer Range: 0 slots to 20 slots, Unit: slots

get_slope() → SignalSlope

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:MEValuation:SLOPe
value: enums.SignalSlope = driver.trigger.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
slope: REDGe | FEDGe REDGe: Rising edge FEDGe: Falling edge

get_smode() → SyncMode

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:MEValuation:SMODe
value: enums.SyncMode = driver.trigger.multiEval.get_smode()
```

Selects the size of the search window for synchronization - normal or enhanced.

return
sync_mode: NORMal | ENHanced

get_source() → str

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:MEValuation:SOURce
value: str = driver.trigger.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
source: string 'Free Run (Fast Sync) ' Free run with synchronization 'Free Run (No Sync) ' Free run without synchronization 'IF Power' Power trigger (received RF power)

get_threshold() → float

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:MEValuation:THReshold
value: float or bool = driver.trigger.multiEval.get_threshold()
```

Defines the trigger threshold for power trigger sources.

return
trig_threshold: (float or boolean) numeric Range: -50 dB to 0 dB, Unit: dB (full scale, i.e. relative to reference level minus external attenuation)

get_timeout() → float

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:MEValuation:TOUT
value: float or bool = driver.trigger.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. This setting has no influence on 'Free Run' measurements.

return
trigger_timeout: (float or boolean) numeric | ON | OFF Range: 0.01 s to 167772.15 s, Unit: s ON | OFF enables or disables the timeout.

set_amode(*acquisition_mode: MevAcquisitionMode*) → None

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:MEValuation:AMODe
driver.trigger.multiEval.set_amode(acquisition_mode = enums.MevAcquisitionMode.
↪SLOT)
```

Selects whether the measurement synchronizes to a slot boundary or to a subframe boundary. The parameter is relevant for 'Free Run (Fast Sync) ' and for list mode measurements with 'Synchronization Mode' = 'Enhanced'.

param acquisition_mode
SLOT | SUBFrame

set_delay(*delay: float*) → None

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:MEValuation:DELay
driver.trigger.multiEval.set_delay(delay = 1.0)
```

Defines a time delaying the start of the measurement relative to the trigger event. This setting has no influence on free run measurements.

param delay
numeric Range: -250E-6 s to 250E-6 s, Unit: s

set_mgap(*min_trig_gap: int*) → None

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:MEValuation:MGAP
driver.trigger.multiEval.set_mgap(min_trig_gap = 1)
```

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
integer Range: 0 slots to 20 slots, Unit: slots

set_slope(*slope: SignalSlope*) → None

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:MEValuation:SLOPe
driver.trigger.multiEval.set_slope(slope = enums.SignalSlope.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 slope
REDGe | FEDGe REDGe: Rising edge FEDGe: Falling edge

set_smode(*sync_mode: SyncMode*) → None

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:MEValuation:SMODe
driver.trigger.multiEval.set_smode(sync_mode = enums.SyncMode.ENHanced)
```

Selects the size of the search window for synchronization - normal or enhanced.

param sync_mode
NORMal | ENHanced

set_source(*source: str*) → None

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:MEValuation:SOURce
driver.trigger.multiEval.set_source(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 source
string 'Free Run (Fast Sync) ' Free run with synchronization 'Free Run (No Sync) ' Free run without synchronization 'IF Power' Power trigger (received RF power)

set_threshold(*trig_threshold: float*) → None

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:MEValuation:THReshold
driver.trigger.multiEval.set_threshold(trig_threshold = 1.0)
```

Defines the trigger threshold for power trigger sources.

param trig_threshold
(float or boolean) numeric Range: -50 dB to 0 dB, Unit: dB (full scale, i.e. relative to reference level minus external attenuation)

set_timeout(*trigger_timeout: float*) → None

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:MEValuation:TOUT
driver.trigger.multiEval.set_timeout(trigger_timeout = 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. This setting has no influence on 'Free Run' measurements.

param trigger_timeout
(float or boolean) numeric | ON | OFF Range: 0.01 s to 167772.15 s, Unit: s ON | OFF enables or disables the timeout.

Cloning the Group

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

Subgroups

6.7.1.1 Catalog

SCPI Command :

```
TRIGger:LTE:MEASurement<Instance>:MEValuation:CATalog:SOURce
```

class CatalogCls

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

get_source() → List[str]

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:MEValuation:CATalog:SOURce
value: List[str] = driver.trigger.multiEval.catalog.get_source()
```

Lists all trigger source values that can be set using method RsCmwLteMeas.Trigger.MultiEval.source.

return

source_list: string Comma-separated list of all supported values. Each value is represented as a string.

6.7.1.2 ListPy

SCPI Commands :

```
TRIGger:LTE:MEASurement<Instance>:MEValuation:LIST:MODE
TRIGger:LTE:MEASurement<Instance>:MEValuation:LIST:NBANdwidth
```

class ListPyCls

ListPy commands group definition. 2 total commands, 0 Subgroups, 2 group commands

get_mode() → ListMode

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:MEValuation:LIST:MODE
value: enums.ListMode = driver.trigger.multiEval.listPy.get_mode()
```

Specifies the trigger mode for list mode measurements. For configuration of retrigger flags, see method RsCmwLteMeas. Configure.MultiEval.ListPy.Segment.Setup.set. For configuration of the global trigger source, see method RsCmwLteMeas. Trigger.MultiEval.source.

return

mode: ONCE | SEGMent ONCE A trigger event is only required to start the measurement. The entire range of segments to be measured is captured without additional trigger event. The global trigger source is used. SEGMent The retrigger flag of each segment is evaluated. It defines whether a trigger event is required and which trigger source is used.

get_nbandwidth() → Nbandwidth

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:MEValuation:LIST:NBANdwidth
value: enums.Nbandwidth = driver.trigger.multiEval.listPy.get_nbandwidth()
```

Selects the trigger evaluation bandwidth for the retrigger source IFPNarrowband. Select the retrigger source via method RsCmwLteMeas.Configure.MultiEval.ListPy.Segment.Setup.set.

return
bandwidth: M010 | M020 | M040 | M080 Evaluation bandwidth 10 MHz to 80 MHz

set_mode(*mode: ListMode*) → None

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:MEValuation:LIST:MODE
driver.trigger.multiEval.listPy.set_mode(mode = enums.ListMode.ONCE)
```

Specifies the trigger mode for list mode measurements. For configuration of retrigger flags, see method RsCmwLteMeas. Configure.MultiEval.ListPy.Segment.Setup.set. For configuration of the global trigger source, see method RsCmwLteMeas. Trigger.MultiEval.source.

param mode
ONCE | SEGMent ONCE A trigger event is only required to start the measurement. The entire range of segments to be measured is captured without additional trigger event. The global trigger source is used. SEGMent The retrigger flag of each segment is evaluated. It defines whether a trigger event is required and which trigger source is used.

set_nbandwidth(*bandwidth: Nbandwidth*) → None

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:MEValuation:LIST:NBANdwidth
driver.trigger.multiEval.listPy.set_nbandwidth(bandwidth = enums.Nbandwidth.
↪M010)
```

Selects the trigger evaluation bandwidth for the retrigger source IFPNarrowband. Select the retrigger source via method RsCmwLteMeas.Configure.MultiEval.ListPy.Segment.Setup.set.

param bandwidth
M010 | M020 | M040 | M080 Evaluation bandwidth 10 MHz to 80 MHz

6.7.2 Prach

SCPI Commands :

```
TRIGger:LTE:MEASurement<Instance>:PRACh:SOURce
TRIGger:LTE:MEASurement<Instance>:PRACh:THReshold
TRIGger:LTE:MEASurement<Instance>:PRACh:SLOPe
TRIGger:LTE:MEASurement<Instance>:PRACh:TOUT
TRIGger:LTE:MEASurement<Instance>:PRACh:MGAP
```

class PrachCls

Prach commands group definition. 6 total commands, 1 Subgroups, 5 group commands

get_mgap() → float

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:PRACh:MGAP
value: float = driver.trigger.prach.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: float Range: 0 s to 1E-3 s, Unit: s

get_slope() → SignalSlope

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:PRACh:SLOPe
value: enums.SignalSlope = driver.trigger.prach.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
slope: REDGe | FEDGe REDGe: Rising edge FEDGe: Falling edge

get_source() → str

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:PRACh:SOURce
value: str = driver.trigger.prach.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
source: string 'IF Power': Power trigger (received RF power)

get_threshold() → float

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:PRACh:THReshold
value: float or bool = driver.trigger.prach.get_threshold()
```

Defines the trigger threshold for power trigger sources.

return
trig_threshold: (float or boolean) numeric Range: -50 dB to 0 dB, Unit: dB (full scale, i.e. relative to reference level minus external attenuation)

get_timeout() → float

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:PRACh:TOUT
value: float or bool = driver.trigger.prach.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. This setting has no influence on 'Free Run' measurements.

return
trigger_timeout: (float or boolean) numeric | ON | OFF Range: 0.01 s to 167772.15 s, Unit: s ON | OFF enables or disables the timeout.

set_mgap(*min_trig_gap: float*) → None

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:PRACh:MGAP
driver.trigger.prach.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
float Range: 0 s to 1E-3 s, Unit: s

set_slope(*slope: SignalSlope*) → None

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:PRACh:SLOPe
driver.trigger.prach.set_slope(slope = enums.SignalSlope.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 slope
REDGe | FEDGe REDGe: Rising edge FEDGe: Falling edge

set_source(*source: str*) → None

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:PRACh:SOURce
driver.trigger.prach.set_source(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 source
string 'IF Power': Power trigger (received RF power)

set_threshold(*trig_threshold: float*) → None

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:PRACh:THReshold
driver.trigger.prach.set_threshold(trig_threshold = 1.0)
```

Defines the trigger threshold for power trigger sources.

param trig_threshold
(float or boolean) numeric Range: -50 dB to 0 dB, Unit: dB (full scale, i.e. relative to reference level minus external attenuation)

set_timeout(*trigger_timeout: float*) → None

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:PRACh:TOUT
driver.trigger.prach.set_timeout(trigger_timeout = 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. This setting has no influence on 'Free Run' measurements.

param trigger_timeout
(float or boolean) numeric | ON | OFF Range: 0.01 s to 167772.15 s, Unit: s ON | OFF enables or disables the timeout.

Cloning the Group

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

Subgroups

6.7.2.1 Catalog

SCPI Command :

```
TRIGger:LTE:MEASurement<Instance>:PRACh:CATalog:SOURce
```

class CatalogCls

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

get_source() → List[str]

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:PRACh:CATalog:SOURce
value: List[str] = driver.trigger.prach.catalog.get_source()
```

Lists all trigger source values that can be set using method RsCmwLteMeas.Trigger.Prach.source.

return
source_list: string Comma-separated list of all supported values. Each value is represented as a string.

6.7.3 Srs

SCPI Commands :

```
TRIGger:LTE:MEASurement<Instance>:SRS:SOURce
TRIGger:LTE:MEASurement<Instance>:SRS:THReshold
TRIGger:LTE:MEASurement<Instance>:SRS:SLOPe
TRIGger:LTE:MEASurement<Instance>:SRS:TOUT
TRIGger:LTE:MEASurement<Instance>:SRS:MGAP
```

class SrsCls

Srs commands group definition. 6 total commands, 1 Subgroups, 5 group commands

get_mgap() → float

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:SRS:MGAP
value: float = driver.trigger.srs.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: float Range: 0 s to 1E-3 s, Unit: s

get_slope() → SignalSlope

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:SRS:SLOPe
value: enums.SignalSlope = driver.trigger.srs.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
slope: REDGe | FEDGe REDGe: Rising edge FEDGe: Falling edge

get_source() → str

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:SRS:SOURce
value: str = driver.trigger.srs.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
source: string 'IF Power': Power trigger (received RF power)

get_threshold() → float

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:SRS:THReshold
value: float or bool = driver.trigger.srs.get_threshold()
```

Defines the trigger threshold for power trigger sources.

return
trig_threshold: (float or boolean) numeric Range: -50 dB to 0 dB, Unit: dB (full scale, i.e. relative to reference level minus external attenuation)

get_timeout() → float

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:SRS:TOUT
value: float or bool = driver.trigger.srs.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. This setting has no influence on 'Free Run' measurements.

return
trigger_timeout: (float or boolean) numeric | ON | OFF Range: 0.01 s to 167772.15 s, Unit: s ON | OFF enables or disables the timeout.

set_mgap(*min_trig_gap: float*) → None

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:SRS:MGAP
driver.trigger.srs.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
float Range: 0 s to 1E-3 s, Unit: s

set_slope(*slope: SignalSlope*) → None

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:SRS:SLOPe
driver.trigger.srs.set_slope(slope = enums.SignalSlope.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 slope
REDGe | FEDGe REDGe: Rising edge FEDGe: Falling edge

set_source(*source: str*) → None

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:SRS:SOURce
driver.trigger.srs.set_source(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 source
string 'IF Power': Power trigger (received RF power)

set_threshold(*trig_threshold: float*) → None

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:SRS:THReshold
driver.trigger.srs.set_threshold(trig_threshold = 1.0)
```

Defines the trigger threshold for power trigger sources.

param trig_threshold
(float or boolean) numeric Range: -50 dB to 0 dB, Unit: dB (full scale, i.e. relative to reference level minus external attenuation)

set_timeout(*trigger_timeout: float*) → None

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:SRS:TOUT
driver.trigger.srs.set_timeout(trigger_timeout = 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. This setting has no influence on 'Free Run' measurements.

param trigger_timeout
(float or boolean) numeric | ON | OFF Range: 0.01 s to 167772.15 s, Unit: s ON | OFF enables or disables the timeout.

Cloning the Group

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

Subgroups

6.7.3.1 Catalog

SCPI Command :

```
TRIGger:LTE:MEASurement<Instance>:SRS:CATalog:SOURce
```

class CatalogCls

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

get_source() → List[str]

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:SRS:CATalog:SOURce
value: List[str] = driver.trigger.srs.catalog.get_source()
```

Lists all trigger source values that can be set using method RsCmwLteMeas.Trigger.Srs.source.

return

source_list: string Comma-separated list of all supported values. Each value is represented as a string.

CHAPTER

SEVEN

RSCMWLTEMEAS UTILITIES

class Utilities

Common utility class. Utility functions common for all types of drivers.

Access snippet: `utils = RsCmwLteMeas.utilities`

property logger: *ScpiLogger*

Scpi Logger interface, see *here*

Access snippet: `logger = RsCmwLteMeas.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 RsCmwLteMeas instance with new VISA session, the session gets a new thread lock. You can assign it to other RsCmwLteMeas sessions in order to share one physical instrument with a multi-thread access.
- If you create new RsCmwLteMeas 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

RSCMWLTEMEAS 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

RSCMWLTEMEAS 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

D

E

F

G

I

L

M

R

S

T

U