RsCmwBluetoothSig

Release 3.8.20.27

Rohde & Schwarz

May 27, 2021

CONTENTS:


CMW

CHAPTER

ONE

GETTING STARTED

1.1 Introduction

RsCmwBluetoothSig 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('3.7.90.32')
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.source_set(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 (in case of big data transfer)

- Multithreading session locking - you can use multiple threads talking to one instrument at the same time

1.2 Installation

RsCmwBluetoothSig 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 RsCmwBluetoothSig`

Option 2 - Installing in Pycharm

- In Pycharm Menu `File->Settings->Project->Project Interpreter` click on the **'+'** button on the bottom left
- Type `RsCmwBluetoothSig` 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 5 easy step for installing the RsCmwBluetoothSig offline:

- Download this python script (**Save target as**): rsinstrument_offline_install.py This installs all the preconditions that the RsCmwBluetoothSig needs.
- Execute the script in your offline computer (supported is python 3.6 or newer)
- Download the RsCmwBluetoothSig package to your computer from the pypi.org: https://pypi.org/project/RsCmwBluetoothSig/#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\RsCmwBluetoothSig-3.8.20.27.tar`

1.3 Finding Available Instruments

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

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

from RsCmwBluetoothSig import *

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

1.4 Initiating Instrument Session

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

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

- Installed RsCmwBluetoothSig Python module Version 3.8.20 or newer from pypi.org
- Installed VISA, for example R&S Visa 5.12 or newer
"""

from RsCmwBluetoothSig import *

# A good practice is to assure that you have a certain minimum version installed
RsCmwBluetoothSig.assert_minimum_version('3.8.20')
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 = RsCmwBluetoothSig(resource_string_1)

idn = driver.utilities.query_str('*IDN?')
print(f"\nHello, I am: '{idn}'")
print(f'RsCmwBluetoothSig 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 2021.

Do not care about specialty of each session kind; RsCmwBluetoothSig 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 = RsCmwBluetoothSig('TCPIP::192.168.56.101::HISLIP', id_query=True, reset=True)
```

- Setting `id_query` to True (default is True) checks, whether your instrument can be used with the RsCmwBluetoothSig 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 RsCmwBluetoothSig allows you to choose which VISA to use:

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

from RsCmwBluetoothSig import *

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

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

from RsCmwBluetoothSig import *

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

More `option_string` tokens are separated by comma:

```
driver = RsCmwBluetoothSig('TCPIP::192.168.56.101::HISLIP', 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 RsCmwBluetoothSig objects:

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

from RsCmwBluetoothSig import *

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

1.5 Plain SCPI Communication

After you have opened the session, you can use the instrument-specific part described in the RsCmwBluetoothSig 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 RsCmwBluetoothSig import *

driver = RsCmwBluetoothSig('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 RsCmwBluetoothSig import *

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

driver = RsCmwBluetoothSig('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
```

1.6 Error Checking

RsCmwBluetoothSig 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.

1.7 Exception Handling

The base class for all the exceptions raised by the RsCmwBluetoothSig 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 RsCmwBluetoothSig import *

```

(continues on next page)

(continued from previous page)

```
driver = None
# Try-catch for initialization. If an error occures, the ResourceError is raised
try:
    driver = RsCmwBluetoothSig('TCPIP::10.112.1.179::HISLIP')
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 RsCmwBluetoothSig exceptions
    print(e.args[0])
    print('Some other RsCmwBluetoothSig 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.

1.8 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 RsCmwBluetoothSig, 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 RsCmwBluetoothSig 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')
```

1.9 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")
```

1.10 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 RsCmwBluetoothSig has this covered. And, this feature is quite universal - not just for big files transfer, but for any data in both directions.

RsCmwBluetoothSig 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 RsCmwBluetoothSig 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 = RsCmwBluetoothSig('TCPIP::192.168.56.101::INSTR')

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

(continues on next page)

(continued from previous page)

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

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

If you start it, you might wonder (or maybe not): why is the `args.total_size = None`? The reason is, in this particular case the RsCmwBluetoothSig 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
```

1.11 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, RsCmwBluetoothSig 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 RsCmwBluetoothSig object
"""

import threading
from RsCmwBluetoothSig import *
```

(continues on next page)

(continued from previous page)

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


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

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

driver.close()
```

Shared instrument session, accessed from multiple threads

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

Try the following example:

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

import threading
from RsCmwBluetoothSig import *


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

(continues on next page)

(continued from previous page)

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

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

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

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

Multiple instrument sessions accessed from multiple threads

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

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

Run the following example:

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

import threading
from RsCmwBluetoothSig import *


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

(continues on next page)

(continued from previous page)

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

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

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

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

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

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

CHAPTER

TWO

REVISION HISTORY

Rohde & Schwarz CMW Base System RsCmwBase instrument driver.

Supported instruments: CMW500, CMW100, CMW270, CMW280

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

Documentation: https://RsCmwBase.readthedocs.io/

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

Currently supported CMW subsystems:

- Base: RsCmwBase
- Global Purpose RF: RsCmwGprfGen, RsCmwGprfMeas
- Bluetooth: RsCmwBluetoothSig, RsCmwBluetoothMeas
- LTE: RsCmwLteSig, RsCmwLteMeas
- CDMA2000: RsCdma2kSig, RsCdma2kMeas
- 1xEVDO: RsCmwEvdoSig, RsCmwEvdoMeas
- WCDMA: RsCmwWcdmaSig, RsCmwWcdmaMeas
- GSM: RsCmwGsmSig, RsCmwGsmMeas
- WLAN: RsCmwWlanSig, RscmwWlanMeas
- DAU: RsCMwDau

In case you require support for more subsystems, please contact our customer support on customersupport@rohde-schwarz.com with the topic "Auto-generated Python drivers" in the email subject. This will speed up the response process

---

Examples: Download the file 'CMW Python instrument drivers' from https://www.rohde-schwarz.com/driver/cmw500_overview/ The zip file contains the examples on how to use these drivers. Remember to adjust the resourceName string to fit your instrument.

Release Notes for the whole RsCmwXXX group:

Latest release notes summary: <INVALID>

Version 3.7.90.39

- <INVALID>

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

THREE

ENUMS

3.1 AddressType

```
# Example value:
value = enums.AddressType.PUBLic
# All values (2x):
PUBLic | RANDom
```

3.2 AddressTypeExt

```
# Example value:
value = enums.AddressTypeExt.PIDentity
# All values (4x):
PIDentity | PUBLic | RANDom | RSIDentity
```

3.3 AfHopingMode

```
# Example value:
value = enums.AfHopingMode.EUT
# All values (3x):
EUT | NORM | USER
```

3.4 AllocMethod

```
# Example value:
value = enums.AllocMethod.LOUDness
# All values (2x):
LOUDness | SNR
```

3.5 AudioChannelMode

```
# Example value:
value = enums.AudioChannelMode.DUAL
# All values (4x):
DUAL | JSTereo | MONO | STEReo
```

3.6 AudioCodec

```
# Example value:
value = enums.AudioCodec.SBC
# All values (1x):
SBC
```

3.7 BaudRate

```
# First value:
value = enums.BaudRate.B110
# Last value:
value = enums.BaudRate.B96K
# All values (24x):
B110 | B115k | B12K | B14K | B19K | B1M | B1M5 | B234k
B24K | B28K | B2M | B300 | B38K | B3M | B3M5 | B460k
B48K | B4M | B500k | B576k | B57K | B600 | B921k | B96K
```

3.8 BbBoard

```
# First value:
value = enums.BbBoard.BBR1
# Last value:
value = enums.BbBoard.SUW44
# All values (140x):
BBR1 | BBR11 | BBR12 | BBR13 | BBR14 | BBR2 | BBR21 | BBR22
BBR23 | BBR24 | BBR3 | BBR31 | BBR32 | BBR33 | BBR34 | BBR4
BBR41 | BBR42 | BBR43 | BBR44 | BBT1 | BBT11 | BBT12 | BBT13
BBT14 | BBT2 | BBT21 | BBT22 | BBT23 | BBT24 | BBT3 | BBT31
BBT32 | BBT33 | BBT34 | BBT4 | BBT41 | BBT42 | BBT43 | BBT44
SUA012 | SUA034 | SUA056 | SUA078 | SUA1 | SUA11 | SUA112 | SUA12
SUA13 | SUA134 | SUA14 | SUA15 | SUA156 | SUA16 | SUA17 | SUA178
SUA18 | SUA2 | SUA21 | SUA212 | SUA22 | SUA23 | SUA234 | SUA24
SUA25 | SUA256 | SUA26 | SUA27 | SUA278 | SUA28 | SUA3 | SUA31
SUA312 | SUA32 | SUA33 | SUA334 | SUA34 | SUA35 | SUA356 | SUA36
SUA37 | SUA378 | SUA38 | SUA4 | SUA41 | SUA412 | SUA42 | SUA43
SUA434 | SUA44 | SUA45 | SUA456 | SUA46 | SUA47 | SUA478 | SUA48
SUA5 | SUA6 | SUA7 | SUA8 | SUU1 | SUU11 | SUU12 | SUU13
```

(continues on next page)

(continued from previous page)

```
SUU14 | SUU2 | SUU21 | SUU22 | SUU23 | SUU24 | SUU3 | SUU31
SUU32 | SUU33 | SUU34 | SUU4 | SUU41 | SUU42 | SUU43 | SUU44
SUW1 | SUW11 | SUW12 | SUW13 | SUW14 | SUW2 | SUW21 | SUW22
SUW23 | SUW24 | SUW3 | SUW31 | SUW32 | SUW33 | SUW34 | SUW4
SUW41 | SUW42 | SUW43 | SUW44
```

3.9 BlockLength

```
# Example value:
value = enums.BlockLength.BL12
# All values (4x):
BL12 | BL16 | BL4 | BL8
```

3.10 BrPacketType

```
# Example value:
value = enums.BrPacketType.DH1
# All values (3x):
DH1 | DH3 | DH5
```

3.11 BurstType

```
# Example value:
value = enums.BurstType.BR
# All values (3x):
BR | EDR | LE
```

3.12 CodingScheme

```
# Example value:
value = enums.CodingScheme.S2
# All values (2x):
S2 | S8
```

3.13 CommProtocol

```
# Example value:
value = enums.CommProtocol.HCI
# All values (2x):
HCI | TWO
```

3.14 ConnectAction

```
# First value:
value = enums.ConnectAction.ADConnect
# Last value:
value = enums.ConnectAction.TMConnect
# All values (21x):
ADConnect | ADENter | ADEXit | AGConnect | AUDConnect | CONNect | DETach | EMConnect
ENAGate | ENEMode | ENHFp | EXAGate | EXEMode | EXHFp | HFPConnect | INQuire
REController | SCONnecting | SINQuiry | STMode | TMConnect
```

3.15 ConnectionActionLe

```
# Example value:
value = enums.ConnectionActionLe.CONNect
# All values (6x):
CONNect | DETach | INQuire | SCONnecting | SINQuiry | TMConnect
```

3.16 ConnectionState

```
# First value:
value = enums.ConnectionState.A2CNnecting
# Last value:
value = enums.ConnectionState.XHASmode
# All values (47x):
A2CNnecting | A2Connected | A2Detaching | A2SCnnected | A2SDetaching | A2SNnecting |␣
↪ACNNecting | ACONected
AENMode | AEXMode | AGCNnecting | AGConnected | CHASmode | CNASmode | CNNecting |␣
↪CONNected
DETaching | DHASmode | ECNNecting | ECONected | ECRunning | EHASmode | ENAGmode | ENEMode
ENHFp | ENHSmode | EXAGmode | EXEMode | EXHFp | EXHSmode | HFCNnecting | HFConnected
HSCNnecting | HSConnected | HSDetaching | INQuiring | OFF | SBY | SCONnecting | SINQuiry
SMCNnecting | SMConnected | SMDetaching | SMIDle | TCNNecting | TCONected | XHASmode
```

3.17 ConTestResult

```
# Example value:
value = enums.ConTestResult.FAIL
# All values (4x):
FAIL | NRUN | PASS | TOUT
```

3.18 CteType

```
# Example value:
value = enums.CteType.AOA1us
# All values (4x):
AOA1us | AOA2us | AOD1us | AOD2us
```

3.19 DataBits

```
# Example value:
value = enums.DataBits.D7
# All values (2x):
D7 | D8
```

3.20 DriftRate

```
# Example value:
value = enums.DriftRate.HDRF
# All values (2x):
HDRF | LDRF
```

3.21 DtxMode

```
# Example value:
value = enums.DtxMode.SINGle
# All values (2x):
SINGle | SPEC
```

3.22 EdrPacketType

```
# Example value:
value = enums.EdrPacketType.E21P
# All values (6x):
E21P | E23P | E25P | E31P | E33P | E35P
```

3.23 EutState

```
# Example value:
value = enums.EutState.FAIL
# All values (2x):
FAIL | OK
```

3.24 HwInterface

```
# Example value:
value = enums.HwInterface.NONE
# All values (3x):
NONE | RS232 | USB
```

3.25 LeDiagState

```
# Example value:
value = enums.LeDiagState.LOADingvec
# All values (4x):
LOADingvec | OFF | ON | VECTorloaded
```

3.26 LeHoppingMode

```
# Example value:
value = enums.LeHoppingMode.ALL
# All values (2x):
ALL | CH2
```

3.27 LePacketType2

```
# Example value:
value = enums.LePacketType2.RFCTe
# All values (2x):
RFCTe | RFPHytest
```

3.28 LePhysicalType

```
# Example value:
value = enums.LePhysicalType.LE1M
# All values (3x):
LE1M | LE2M | LELR
```

3.29 LeRangePaternType

```
# Example value:
value = enums.LeRangePaternType.ALL0
# All values (6x):
ALL0 | ALL1 | ALT | P11 | P44 | PRBS9
```

3.30 LeSignalingState

```
# First value:
value = enums.LeSignalingState.CMR
# Last value:
value = enums.LeSignalingState.TXRunning
# All values (12x):
CMR | IDLE | OFF | RCOM | RXRunning | SPCM | SPRX | SPTX
STCM | STRX | STTX | TXRunning
```

3.31 LogCategory

```
# Example value:
value = enums.LogCategory.CONTinue
# All values (4x):
CONTinue | ERRor | INFO | WARNing
```

3.32 ModIndexType

```
# Example value:
value = enums.ModIndexType.STAB
# All values (2x):
STAB | STAN
```

3.33 OperatingMode

```
# Example value:
value = enums.OperatingMode.AUDio
# All values (6x):
AUDio | CNTest | ECMode | LETMode | PROFiles | RFTest
```

3.34 PacketTypeEsco

```
# First value:
value = enums.PacketTypeEsco._2EV3
# Last value:
value = enums.PacketTypeEsco.HV3
# All values (10x):
_2EV3 | _2EV5 | _3EV3 | _3EV5 | EV3 | EV4 | EV5 | HV1
HV2 | HV3
```

3.35 PacketTypeSco

```
# Example value:
value = enums.PacketTypeSco.HV1
# All values (3x):
HV1 | HV2 | HV3
```

3.36 PageScanMode

```
# Example value:
value = enums.PageScanMode._0X00
# All values (4x):
_0X00 | _0X01 | _0X02 | _0X03
```

3.37 PageScanPeriodMode

```
# Example value:
value = enums.PageScanPeriodMode.P0
# All values (3x):
P0 | P1 | P2
```

3.38 Parity

```
# Example value:
value = enums.Parity.EVEN
# All values (3x):
EVEN | NONE | ODD
```

3.39 PowerChange

```
# Example value:
value = enums.PowerChange.DOWN
# All values (4x):
DOWN | MAX | NNE | UP
```

3.40 PowerControl

```
# Example value:
value = enums.PowerControl.DOWN
# All values (3x):
DOWN | MAX | UP
```

3.41 PowerControlMode

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

3.42 PowerFlag

```
# Example value:
value = enums.PowerFlag.MAX
# All values (3x):
MAX | MIN | NONE
```

3.43 PowerMinMax

```
# Example value:
value = enums.PowerMinMax.CHANged
# All values (5x):
CHANged | MAX | MIN | NNM | NOTS
```

3.44 PriorityRole

```
# Example value:
value = enums.PriorityRole.MASTer
# All values (2x):
MASTer | SLAVe
```

3.45 ProfileRole

```
# Example value:
value = enums.ProfileRole.ADGate
# All values (3x):
ADGate | ASINk | HNDFree
```

3.46 Protocol

```
# Example value:
value = enums.Protocol.CTSRts
# All values (3x):
CTSRts | NONE | XONXoff
```

3.47 PsrMode

```
# Example value:
value = enums.PsrMode.R0
# All values (3x):
R0 | R1 | R2
```

3.48 Repeat

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

3.49 ResourceState

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

3.50 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
```

3.51 RxConnector

```
# First value:
value = enums.RxConnector.I11I
# Last value:
value = enums.RxConnector.RH8
# All values (154x):
I11I | I13I | I15I | I17I | I21I | I23I | I25I | I27I
I31I | I33I | I35I | I37I | I41I | I43I | I45I | I47I
IF1 | IF2 | IF3 | IQ1I | IQ3I | IQ5I | IQ7I | R11
R11C | R12 | R12C | R12I | R13 | R13C | R14 | R14C
R14I | R15 | R16 | R17 | R18 | R21 | R21C | R22
```

(continues on next page)

(continued from previous page)

```
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 | RF8 | RFAC
RFBC | RFBI | RG1 | RG2 | RG3 | RG4 | RG5 | RG6
RG7 | RG8 | RH1 | RH2 | RH3 | RH4 | RH5 | RH6
RH7 | RH8
```

3.52 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
```

3.53 SamplingFrequency

```
# Example value:
value = enums.SamplingFrequency.SF16
# All values (4x):
SF16 | SF32 | SF441 | SF48
```

3.54 SecurityMode

```
# Example value:
value = enums.SecurityMode.SEC2
# All values (2x):
SEC2 | SEC3
```

3.55 SequenceNumbering

```
# Example value:
value = enums.SequenceNumbering.NORM
# All values (2x):
NORM | TEST
```

3.56 SignalingCmwRole

```
# Example value:
value = enums.SignalingCmwRole.CENTral
# All values (2x):
CENTral | PERipheral
```

3.57 SignalingStandard

```
# Example value:
value = enums.SignalingStandard.CLASsic
# All values (2x):
CLASsic | LESignaling
```

3.58 SignalingState

```
# First value:
value = enums.SignalingState.CNNecting
# Last value:
value = enums.SignalingState.TCONected
# All values (9x):
CNNecting | CONNected | CPOWer | DETaching | INQuiring | OFF | SBY | TCNNecting
TCONected
```

3.59 SpeechCode

```
# Example value:
value = enums.SpeechCode.ALAW
# All values (4x):
ALAW | CVSD | MSBC | ULAW
```

3.60 StopBits

```
# Example value:
value = enums.StopBits.S1
# All values (2x):
S1 | S2
```

3.61 SubBands

```
# Example value:
value = enums.SubBands.SB4
# All values (2x):
SB4 | SB8
```

3.62 SymbolTimeError

```
# Example value:
value = enums.SymbolTimeError.NEG20
# All values (3x):
NEG20 | OFF | POS20
```

3.63 SymbolTimeErrorLe

```
# Example value:
value = enums.SymbolTimeErrorLe.NEG50
# All values (3x):
NEG50 | OFF | POS50
```

3.64 TestMode

```
# Example value:
value = enums.TestMode.LOOPback
# All values (2x):
LOOPback | TXTest
```

3.65 TestVector

```
# First value:
value = enums.TestVector.INITstack
# Last value:
value = enums.TestVector.TV9
# All values (47x):
INITstack | RELoadstack | TV0 | TV1 | TV10 | TV11 | TV12 | TV13
TV14 | TV15 | TV16 | TV17 | TV18 | TV19 | TV2 | TV20
TV21 | TV22 | TV23 | TV24 | TV25 | TV26 | TV27 | TV28
TV29 | TV3 | TV30 | TV31 | TV32 | TV33 | TV34 | TV35
TV36 | TV37 | TV38 | TV39 | TV4 | TV40 | TV41 | TV42
TV43 | TV44 | TV5 | TV6 | TV7 | TV8 | TV9
```

3.66 TxConnector

```
# First value:
value = enums.TxConnector.I120
# Last value:
value = enums.TxConnector.RH18
# All values (77x):
I120 | I140 | I160 | I180 | I220 | I240 | I260 | I280
I320 | I340 | I360 | I380 | I420 | I440 | I460 | I480
IF1 | IF2 | IF3 | IQ20 | IQ40 | IQ60 | IQ80 | R118
R1183 | R1184 | R11C | R110 | R1103 | R1104 | R12C | R13C
R130 | R14C | R214 | R218 | R21C | R210 | R22C | R23C
R230 | R24C | R258 | R318 | R31C | R310 | R32C | R33C
R330 | R34C | R418 | R41C | R410 | R42C | R43C | R430
R44C | RA18 | RB14 | RB18 | RC18 | RD18 | RE18 | RF18
RF1C | RF10 | RF2C | RF3C | RF30 | RF4C | RF5C | RF6C
RFAC | RFA0 | RFBC | RG18 | RH18
```

3.67 TxConverter

```
# First value:
value = enums.TxConverter.ITX1
# Last value:
value = enums.TxConverter.TX44
# All values (40x):
ITX1 | ITX11 | ITX12 | ITX13 | ITX14 | ITX2 | ITX21 | ITX22
ITX23 | ITX24 | ITX3 | ITX31 | ITX32 | ITX33 | ITX34 | ITX4
ITX41 | ITX42 | ITX43 | ITX44 | TX1 | TX11 | TX12 | TX13
TX14 | TX2 | TX21 | TX22 | TX23 | TX24 | TX3 | TX31
TX32 | TX33 | TX34 | TX4 | TX41 | TX42 | TX43 | TX44
```

3.68 VoiceLinkType

```
# Example value:
value = enums.VoiceLinkType.ESCO
# All values (2x):
ESCO | SCO
```

CHAPTER

FOUR

REPCAPS

4.1 Instance (Global)

```
# Setting:
driver.repcap_instance_set(repcap.Instance.Inst1)
# Values (4x):
Inst1 | Inst2 | Inst3 | Inst4
```

4.2 CommSettings

```
# First value:
value = repcap.CommSettings.Hw1
# Values (4x):
Hw1 | Hw2 | Hw3 | Hw4
```

4.3 HardwareIntf

```
# First value:
value = repcap.HardwareIntf.Intf1
# Values (4x):
Intf1 | Intf2 | Intf3 | Intf4
```

4.4 UsbSettings

```
# First value:
value = repcap.UsbSettings.Sett1
# Values (4x):
Sett1 | Sett2 | Sett3 | Sett4
```

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('3.7.90.32')
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:
↪{event_args.message}')
```

(continues on next page)

(continued from previous page)

```
# 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.source_set(enums.SourceIntExt.INTernal)

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

CHAPTER

SIX

INDEX

CHAPTER

SEVEN

RSCMWBLUETOOTHSIG API STRUCTURE

Global RepCaps

```
driver = RsCmwBluetoothSig('TCPIP::192.168.2.101::HISLIP')
# Instance range: Inst1 .. Inst4
rc = driver.repcap_instance_get()
driver.repcap_instance_set(repcap.Instance.Inst1)
```

class RsCmwBluetoothSig(*resource_name: str*, *id_query: bool = True*, *reset: bool = False*, *options: Optional[str] = None*, *direct_session: Optional[object] = None*)

621 total commands, 10 Sub-groups, 0 group commands

Initializes new RsCmwBluetoothSig 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=ni' - forces usage of National Instruments Visa
- 'QueryInstrumentStatus = False' - same as driver.utilities.instrument_status_checking = False
- 'DriverSetup=(WriteDelay = 20, ReadDelay = 5)' - Introduces delay of 20ms before each write and 5ms before each read
- 'DriverSetup=(OpcWaitMode = OpcQuery)' - mode for all the opc-synchronised write/reads. Other modes: StbPolling, StbPollingSlow, StbPollingSuperSlow
- 'DriverSetup=(AddTermCharToWriteBinBLock = True)' - Adds one additional LF to the end of the binary data (some instruments require that)
- 'DriverSetup=(AssureWriteWithTermChar = True)' - Makes sure each command/query is terminated with termination character. Default: Interface dependent
- 'DriverSetup=(TerminationCharacter = 'x')' - Sets the termination character for reading. Default: '<LF>' (LineFeed)
- 'DriverSetup=(IoSegmentSize = 10E3)' - Maximum size of one write/read segment. If transferred data is bigger, it is split to more segments
- 'DriverSetup=(OpcTimeout = 10000)' - same as driver.utilities.opc_timeout = 10000
- 'DriverSetup=(VisaTimeout = 5000)' - same as driver.utilities.visa_timeout = 5000

- 'DriverSetup=(ViClearExeMode = 255)' - Binary combination where 1 means performing viClear() on a certain interface as the very first command in init
- 'DriverSetup=(OpcQueryAfterWrite = True)' - same as driver.utilities.opc_query_after_write = True

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.

close() → None

Closes the active RsCmwBluetoothSig session.

classmethod from_existing_session(*session: object*, *options: Optional[str] = None*) → RsCmwBluetoothSig

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

Parameters

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

get_session_handle() → object

Returns the underlying session handle.

static list_resources(*expression: str = '?*::INSTR'*, *visa_select: Optional[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: '@ni', '@rs'

restore_all_repcaps_to_default() → None

Sets all the Group and Global repcaps to their initial values

Subgroups

7.1 Configure

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:OPMode
CONFigure:BLUetooth:SIGNaling<Instance>:CPRotocol
CONFigure:BLUetooth:SIGNaling<Instance>:STANdard
```

class Configure

Configure commands group definition. 396 total commands, 12 Sub-groups, 3 group commands

get_cprotocol() → RsCmwBluetoothSig.enums.CommProtocol

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CPRotocol
value: enums.CommProtocol = driver.configure.get_cprotocol()
```

Specifies the communication protocol for direct test mode.

return comm_protocol: No help available

get_op_mode() → RsCmwBluetoothSig.enums.OperatingMode

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:OPMode
value: enums.OperatingMode = driver.configure.get_op_mode()
```

Specifies operating mode of R&S CMW.

return operating_mode: CNTest | RFTest | ECMode | PROFiles | AUDio | LETMode CNTest: connection test for BR/EDR or LE (OTA) RFTest: test mode for BR/EDR or direct test for LE ECMode: echo mode for BR/EDR PROFiles: profiles for BR/EDR AUDio: audio mode for BR/EDR LETMode: LE test mode (OTA)

get_standard() → RsCmwBluetoothSig.enums.SignalingStandard

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:STANdard
value: enums.SignalingStandard = driver.configure.get_standard()
```

Selects classic (BR/EDR) or low energy (LE) bursts.

return sig_std: CLASSic | LESignaling

set_cprotocol(*comm_protocol: RsCmwBluetoothSig.enums.CommProtocol*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CPRotocol
driver.configure.set_cprotocol(comm_protocol = enums.CommProtocol.HCI)
```

Specifies the communication protocol for direct test mode.

param comm_protocol HCI | TWO HCI or two-wire UART interface

set_op_mode(*operating_mode: RsCmwBluetoothSig.enums.OperatingMode*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:OPMode
driver.configure.set_op_mode(operating_mode = enums.OperatingMode.AUDio)
```

Specifies operating mode of R&S CMW.

param operating_mode CNTest | RFTest | ECMode | PROFiles | AUDio | LETMode
CNTest: connection test for BR/EDR or LE (OTA) RFTest: test mode for BR/EDR or direct test for LE ECMode: echo mode for BR/EDR PROFiles: profiles for BR/EDR AUDio: audio mode for BR/EDR LETMode: LE test mode (OTA)

set_standard(*sig_std: RsCmwBluetoothSig.enums.SignalingStandard*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:STANdard
driver.configure.set_standard(sig_std = enums.SignalingStandard.CLASsic)
```

Selects classic (BR/EDR) or low energy (LE) bursts.

param sig_std CLASsic | LESignaling

Cloning the Group

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

Subgroups

7.1.1 Delay

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:DELay:PTIMeout
CONFigure:BLUetooth:SIGNaling<Instance>:DELay:TMODe
```

class Delay

Delay commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

get_ptimeout() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:DELay:PTIMeout
value: int = driver.configure.delay.get_ptimeout()
```

Sets delay for the processing of HCI commands. If set to 100 ms, maximally one HCI command is processed every 100 ms.

return poll_timeout: integer Range: 1 ms to 100 ms , Unit: ms

get_tmode() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:DELay:TMODe
value: int = driver.configure.delay.get_tmode()
```

Specifies delay for test mode activation. The delay is applied after acknowledgment from EUT that it has received activate test mode command.

return act_test_delay: integer Range: 1 ms to 100 ms , Unit: ms

set_ptimeout(*poll_timeout: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:DELay:PTIMeout
driver.configure.delay.set_ptimeout(poll_timeout = 1)
```

Sets delay for the processing of HCI commands. If set to 100 ms, maximally one HCI command is processed every 100 ms.

param poll_timeout integer Range: 1 ms to 100 ms , Unit: ms

set_tmode(*act_test_delay: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:DELay:TMODe
driver.configure.delay.set_tmode(act_test_delay = 1)
```

Specifies delay for test mode activation. The delay is applied after acknowledgment from EUT that it has received activate test mode command.

param act_test_delay integer Range: 1 ms to 100 ms , Unit: ms

7.1.2 Tmode

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:TMODe:LENergy
CONFigure:BLUetooth:SIGNaling<Instance>:TMODe
```

class Tmode

Tmode commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

get_low_energy() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:TMODe:LENergy
value: bool = driver.configure.tmode.get_low_energy()
```

Enables or disables LE test mode at the R&S CMW.

return enable_test_mode: OFF | ON

get_value() → RsCmwBluetoothSig.enums.TestMode

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:TMODe
value: enums.TestMode = driver.configure.tmode.get_value()
```

Selects the test mode that the EUT enters in a test mode connection.

return test_mode: LOOPback | TXTest LOOPback: BR/EDR loopback test mode TXTest: BR/EDR transmitter test mode

set_low_energy(*enable_test_mode: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:TMODe:LENergy
driver.configure.tmode.set_low_energy(enable_test_mode = False)
```

Enables or disables LE test mode at the R&S CMW.

param enable_test_mode OFF | ON

set_value(*test_mode: RsCmwBluetoothSig.enums.TestMode*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:TMODe
driver.configure.tmode.set_value(test_mode = enums.TestMode.LOOPback)
```

Selects the test mode that the EUT enters in a test mode connection.

param test_mode LOOPback | TXTest LOOPback: BR/EDR loopback test mode TXTest: BR/EDR transmitter test mode

7.1.3 Audio

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:AUDio:PRFRole
CONFigure:BLUetooth:SIGNaling<Instance>:AUDio:CMWRole
```

class Audio

Audio commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

get_cmw_role() → RsCmwBluetoothSig.enums.PriorityRole

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:AUDio:CMWRole
value: enums.PriorityRole = driver.configure.audio.get_cmw_role()
```

No command help available

return cmw_role: No help available

get_prf_role() → RsCmwBluetoothSig.enums.ProfileRole

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:AUDio:PRFRole
value: enums.ProfileRole = driver.configure.audio.get_prf_role()
```

Specifies the audio profile role of the EUT.

return profile_role: HNDFree | ADGate | ASINk Hands free, hands free - audio gateway, A2DP sink

set_cmw_role(*cmw_role: RsCmwBluetoothSig.enums.PriorityRole*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:AUDio:CMWRole
driver.configure.audio.set_cmw_role(cmw_role = enums.PriorityRole.MASTer)
```

No command help available

param cmw_role No help available

set_prf_role(*profile_role: RsCmwBluetoothSig.enums.ProfileRole*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:AUDio:PRFRole
driver.configure.audio.set_prf_role(profile_role = enums.ProfileRole.ADGate)
```

Specifies the audio profile role of the EUT.

param profile_role HNDFree | ADGate | ASINk Hands free, hands free - audio gateway, A2DP sink

7.1.4 Connection

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:BTYPe
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:DELay
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:WHITening
```

class Connection

Connection commands group definition. 92 total commands, 24 Sub-groups, 3 group commands

get_btype() → RsCmwBluetoothSig.enums.BurstType

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:BTYPe
value: enums.BurstType = driver.configure.connection.get_btype()


    INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant␣
↪for:

    - BR/EDR in test mode
    - LE in direct test mode

    :return: burst_type: BR | EDR | LE BR: 'Basic Rate' EDR: 'Enhanced Data Rate
↪' LE: 'Low Energy'
```

get_delay() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:DELay
value: bool = driver.configure.connection.get_delay()
```

No command help available

return delay: No help available

get_whitening() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:WHITening
value: bool = driver.configure.connection.get_whitening()
```

Sets whether the EUT has to transmit ACL packets scrambled with a particular data sequence in a loopback mode.

return whitening: OFF | ON

set_btype(*burst_type: RsCmwBluetoothSig.enums.BurstType*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:BTYPe
driver.configure.connection.set_btype(burst_type = enums.BurstType.BR)


    INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant␣
↪for:

    - BR/EDR in test mode
    - LE in direct test mode

    :param burst_type: BR | EDR | LE BR: 'Basic Rate' EDR: 'Enhanced Data Rate'␣
↪LE: 'Low Energy'
```

set_delay(*delay: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:DELay
driver.configure.connection.set_delay(delay = False)
```

No command help available

param delay No help available

set_whitening(*whitening: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:WHITening
driver.configure.connection.set_whitening(whitening = False)
```

Sets whether the EUT has to transmit ACL packets scrambled with a particular data sequence in a loopback mode.

param whitening OFF | ON

Cloning the Group

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

Subgroups

7.1.4.1 Audio

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:SECMode
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:VLINk
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:PINCode
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:CODec
```

class Audio

Audio commands group definition. 17 total commands, 3 Sub-groups, 4 group commands

get_codec() → RsCmwBluetoothSig.enums.SpeechCode

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:CODec
value: enums.SpeechCode = driver.configure.connection.audio.get_codec()
```

Specifies the codec to be used for synchronous connection-oriented audio connections.

return codec: CVSD | ALAW | ULAW | MSBC CVSD: continuously variable slope delta codec (8 kHz - SCO link) ALAW: A-law coding (8 kHz - SCO link) ULAW: -law coding (8 kHz - SCO link) mSBC: modified subband coding (16 kHz - eSCO link)

get_pin_code() → str

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:PINCode
value: str = driver.configure.connection.audio.get_pin_code()
```

Specifies PIN code for audio profile tests.

return pin_code: string

get_sec_mode() → RsCmwBluetoothSig.enums.SecurityMode

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:SECMode
value: enums.SecurityMode = driver.configure.connection.audio.get_sec_mode()
```

Specifies security mode for audio tests.

return security_mode: SEC2 | SEC3

get_vlink() → RsCmwBluetoothSig.enums.VoiceLinkType

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:VLINk
value: enums.VoiceLinkType = driver.configure.connection.audio.get_vlink()
```

No command help available

return voice_link: No help available

set_codec(*codec: RsCmwBluetoothSig.enums.SpeechCode*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:CODec
driver.configure.connection.audio.set_codec(codec = enums.SpeechCode.ALAW)
```

Specifies the codec to be used for synchronous connection-oriented audio connections.

param codec CVSD | ALAW | ULAW | MSBC CVSD: continuously variable slope delta codec (8 kHz - SCO link) ALAW: A-law coding (8 kHz - SCO link) ULAW: -law coding (8 kHz - SCO link) mSBC: modified subband coding (16 kHz - eSCO link)

set_pin_code(*pin_code: str*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:PINCode
driver.configure.connection.audio.set_pin_code(pin_code = '1')
```

Specifies PIN code for audio profile tests.

param pin_code string

set_sec_mode(*security_mode: RsCmwBluetoothSig.enums.SecurityMode*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:SECMode
driver.configure.connection.audio.set_sec_mode(security_mode = enums.
↪SecurityMode.SEC2)
```

Specifies security mode for audio tests.

param security_mode SEC2 | SEC3

set_vlink(*voice_link: RsCmwBluetoothSig.enums.VoiceLinkType*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:VLINk
driver.configure.connection.audio.set_vlink(voice_link = enums.VoiceLinkType.
↪ESCO)
```

No command help available

param voice_link No help available

Cloning the Group

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

Subgroups

7.1.4.1.1 VolControl

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:VOLControl:MICGain
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:VOLControl:SPEaker
```

class VolControl

VolControl commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

get_mic_gain() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:AUDio:VOLControl:MICGain
value: int = driver.configure.connection.audio.volControl.get_mic_gain()
```

Controls microphone gain for audio tests.

return microphone_gain: numeric Range: 0 to 15

get_speaker() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:AUDio:VOLControl:SPEaker
value: int = driver.configure.connection.audio.volControl.get_speaker()
```

Controls speaker volume for audio tests.

return speaker_volume: numeric Range: 0 to 15

set_mic_gain(*microphone_gain: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:AUDio:VOLControl:MICGain
driver.configure.connection.audio.volControl.set_mic_gain(microphone_gain = 1)
```

Controls microphone gain for audio tests.

param microphone_gain numeric Range: 0 to 15

set_speaker(*speaker_volume: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:AUDio:VOLControl:SPEaker
driver.configure.connection.audio.volControl.set_speaker(speaker_volume = 1)
```

Controls speaker volume for audio tests.

param speaker_volume numeric Range: 0 to 15

7.1.4.1.2 Hfp

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:HFP:CASTartup
```

class Hfp

Hfp commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_castartup() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:HFP:CASTartup
value: bool = driver.configure.connection.audio.hfp.get_castartup()
```

Enables the indication of the active call to the EUT for the audio profile role hands free.

return call_activeat_startup: OFF | ON

set_castartup(*call_activeat_startup: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:HFP:CASTartup
driver.configure.connection.audio.hfp.set_castartup(call_activeat_startup =␣
↪False)
```

Enables the indication of the active call to the EUT for the audio profile role hands free.

param call_activeat_startup OFF | ON

7.1.4.1.3 A2Dp

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:A2DP:ACCSlave
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:A2DP:BITRate
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:A2DP:MAXBitpool
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:A2DP:MINBitpool
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:A2DP:ALCMethod
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:A2DP:SUBBands
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:A2DP:BLKLength
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:A2DP:CHMode
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:A2DP:SMPFrequency
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:A2DP:CODec
```

class A2Dp

A2Dp commands group definition. 10 total commands, 0 Sub-groups, 10 group commands

get_acc_slave() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:A2DP:ACCSlave
value: bool = driver.configure.connection.audio.a2Dp.get_acc_slave()
```

Allows the EUT to take control of the establishment of the A2DP connection when the R&S CMW acts as the slave.

return assume_acceptor_role_in_slave_mode: OFF | ON

get_alc_method() → RsCmwBluetoothSig.enums.AllocMethod

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:A2DP:ALCMethod
value: enums.AllocMethod = driver.configure.connection.audio.a2Dp.get_alc_
↪method()
```

Defines the algorithm used to calculate the no. of allocated bits to represent each subband sample.

return allocation_method: LOUDness | SNR

get_bitrate() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:A2DP:BITRate
value: int = driver.configure.connection.audio.a2Dp.get_bitrate()
```

Queries the bit rate calculated from the A2DP audio link parameters.

return bitrate: decimal Unit: bit/s

get_blk_length() → RsCmwBluetoothSig.enums.BlockLength

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:A2DP:BLKLength
value: enums.BlockLength = driver.configure.connection.audio.a2Dp.get_blk_
↪length()
```

Specifies the number of blocks of audio samples that are encoded in a single SBC frame.

return block_length: BL4 | BL8 | BL12 | BL16 4, 8, 12, 16 blocks

get_chmode() → RsCmwBluetoothSig.enums.AudioChannelMode

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:A2DP:CHMode
value: enums.AudioChannelMode = driver.configure.connection.audio.a2Dp.get_
↪chmode()
```

Specifies channel mode.

return channel_mode: MONO | DUAL | STEReo | JSTereo Mono, dual, stereo, joint stereo

get_codec() → RsCmwBluetoothSig.enums.AudioCodec

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:A2DP:CODec
value: enums.AudioCodec = driver.configure.connection.audio.a2Dp.get_codec()
```

Specifies A2DP codec.

return codec: SBC Only subband coding is supported

get_max_bit_pool() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:A2DP:MAXBitpool
value: int = driver.configure.connection.audio.a2Dp.get_max_bit_pool()
```

Specifies maximum bitpool value.

return maximum_bitpool: numeric Range: 8 to 250

get_min_bit_pool() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:A2DP:MINBitpool
value: int = driver.configure.connection.audio.a2Dp.get_min_bit_pool()
```

Specifies minimum bitpool value.

return minimum_bitpool: numeric Range: 2 to 18

get_smp_frequency() → RsCmwBluetoothSig.enums.SamplingFrequency

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:AUDio:A2DP:SMPFrequency
value: enums.SamplingFrequency = driver.configure.connection.audio.a2Dp.get_smp_
↪frequency()
```

Specifies the sampling frequency.

return sampling_frequency: SF16 | SF32 | SF441 | SF48 16 kHz, 32 kHz, 44.1 kHz, 48 kHz

get_sub_bands() → RsCmwBluetoothSig.enums.SubBands

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:A2DP:SUBBands
value: enums.SubBands = driver.configure.connection.audio.a2Dp.get_sub_bands()
```

Specifies the number of subbands used by generated signal.

return sub_bands: SB4 | SB8 Subband 4 or 8

set_acc_slave(*assume_acceptor_role_in_slave_mode: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:A2DP:ACCSlave
driver.configure.connection.audio.a2Dp.set_acc_slave(assume_acceptor_role_in_
↪slave_mode = False)
```

Allows the EUT to take control of the establishment of the A2DP connection when the R&S CMW acts as the slave.

param assume_acceptor_role_in_slave_mode OFF | ON

set_alc_method(*allocation_method: RsCmwBluetoothSig.enums.AllocMethod*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:A2DP:ALCMethod
driver.configure.connection.audio.a2Dp.set_alc_method(allocation_method = enums.
↪AllocMethod.LOUDness)
```

Defines the algorithm used to calculate the no. of allocated bits to represent each subband sample.

param allocation_method LOUDness | SNR

set_blk_length(*block_length: RsCmwBluetoothSig.enums.BlockLength*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:A2DP:BLKLength
driver.configure.connection.audio.a2Dp.set_blk_length(block_length = enums.
↪BlockLength.BL12)
```

Specifies the number of blocks of audio samples that are encoded in a single SBC frame.

param block_length BL4 | BL8 | BL12 | BL16 4, 8, 12, 16 blocks

set_chmode(*channel_mode: RsCmwBluetoothSig.enums.AudioChannelMode*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:A2DP:CHMode
driver.configure.connection.audio.a2Dp.set_chmode(channel_mode = enums.
↪AudioChannelMode.DUAL)
```

Specifies channel mode.

param channel_mode MONO | DUAL | STEReo | JSTereo Mono, dual, stereo, joint stereo

set_codec(*codec: RsCmwBluetoothSig.enums.AudioCodec*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:A2DP:CODec
driver.configure.connection.audio.a2Dp.set_codec(codec = enums.AudioCodec.SBC)
```

Specifies A2DP codec.

param codec SBC Only subband coding is supported

set_max_bit_pool(*maximum_bitpool: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:A2DP:MAXBitpool
driver.configure.connection.audio.a2Dp.set_max_bit_pool(maximum_bitpool = 1)
```

Specifies maximum bitpool value.

param maximum_bitpool numeric Range: 8 to 250

set_min_bit_pool(*minimum_bitpool: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:A2DP:MINBitpool
driver.configure.connection.audio.a2Dp.set_min_bit_pool(minimum_bitpool = 1)
```

Specifies minimum bitpool value.

param minimum_bitpool numeric Range: 2 to 18

set_smp_frequency(*sampling_frequency: RsCmwBluetoothSig.enums.SamplingFrequency*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:AUDio:A2DP:SMPFrequency
driver.configure.connection.audio.a2Dp.set_smp_frequency(sampling_frequency =␣
↪enums.SamplingFrequency.SF16)
```

Specifies the sampling frequency.

param sampling_frequency SF16 | SF32 | SF441 | SF48 16 kHz, 32 kHz, 44.1 kHz, 48 kHz

set_sub_bands(*sub_bands: RsCmwBluetoothSig.enums.SubBands*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:A2DP:SUBBands
driver.configure.connection.audio.a2Dp.set_sub_bands(sub_bands = enums.SubBands.
↪SB4)
```

Specifies the number of subbands used by generated signal.

param sub_bands SB4 | SB8 Subband 4 or 8

7.1.4.2 Packets

class Packets

Packets commands group definition. 21 total commands, 5 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.4.2.1 Ptype

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PACKets:PTYPe:SCO
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PACKets:PTYPe:ESCO
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PACKets:PTYPe:BRATe
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PACKets:PTYPe:EDRate
```

class Ptype

Ptype commands group definition. 7 total commands, 1 Sub-groups, 4 group commands

get_brate() → RsCmwBluetoothSig.enums.BrPacketType

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PACKets:PTYPe:BRATe
value: enums.BrPacketType = driver.configure.connection.packets.ptype.get_
↪brate()
```

Sets the BR packet type.

return packet_type: DH1 | DH3 | DH5 Data – high rate packet carrying information bytes plus a 16-bit CRC code, see table below.

get_edrate() → RsCmwBluetoothSig.enums.EdrPacketType

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PACKets:PTYPe:EDRate
value: enums.EdrPacketType = driver.configure.connection.packets.ptype.get_
↪edrate()
```

Sets the EDR packet type.

return packet_type: E21P | E23P | E25P | E31P | E33P | E35P Data – high rate packet carrying information bytes plus a 16-bit CRC code, see table below.

get_esco() → RsCmwBluetoothSig.enums.PacketTypeEsco

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PACKets:PTYPe:ESCO
value: enums.PacketTypeEsco = driver.configure.connection.packets.ptype.get_
↪esco()
```

Specifies the type of the LE test packet. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

return packet_type: RFPHytest | RFCTe 'RFPHytest': test packet according to Bluetooth specification up to version 5.0 'RFCTe': test packet with CTE according to Bluetooth specification version 5.1

get_sco() → RsCmwBluetoothSig.enums.PacketTypeSco

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PACKets:PTYPe:SCO
value: enums.PacketTypeSco = driver.configure.connection.packets.ptype.get_sco()
```

Specifies the type of the LE test packet. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

return packet_type: RFPHytest | RFCTe 'RFPHytest': test packet according to Bluetooth specification up to version 5.0 'RFCTe': test packet with CTE according to Bluetooth specification version 5.1

set_brate(*packet_type: RsCmwBluetoothSig.enums.BrPacketType*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PACKets:PTYPe:BRATe
driver.configure.connection.packets.ptype.set_brate(packet_type = enums.
↪BrPacketType.DH1)
```

Sets the BR packet type.

param packet_type DH1 | DH3 | DH5 Data – high rate packet carrying information bytes plus a 16-bit CRC code, see table below.

set_edrate(*packet_type: RsCmwBluetoothSig.enums.EdrPacketType*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PACKets:PTYPe:EDRate
driver.configure.connection.packets.ptype.set_edrate(packet_type = enums.
↪EdrPacketType.E21P)
```

Sets the EDR packet type.

param packet_type E21P | E23P | E25P | E31P | E33P | E35P Data – high rate packet carrying information bytes plus a 16-bit CRC code, see table below.

set_esco(*packet_type: RsCmwBluetoothSig.enums.PacketTypeEsco*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PACKets:PTYPe:ESCO
driver.configure.connection.packets.ptype.set_esco(packet_type = enums.
↪PacketTypeEsco._2EV3)
```

Specifies the type of the LE test packet. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

param packet_type RFPHytest | RFCTe 'RFPHytest': test packet according to Bluetooth specification up to version 5.0 'RFCTe': test packet with CTE according to Bluetooth specification version 5.1

set_sco(*packet_type: RsCmwBluetoothSig.enums.PacketTypeSco*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PACKets:PTYPe:SCO
driver.configure.connection.packets.ptype.set_sco(packet_type = enums.
↪PacketTypeSco.HV1)
```

Specifies the type of the LE test packet. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

param packet_type RFPHytest | RFCTe 'RFPHytest': test packet according to Bluetooth specification up to version 5.0 'RFCTe': test packet with CTE according to Bluetooth specification version 5.1

Cloning the Group

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

Subgroups

7.1.4.2.1.1 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PACKets:PTYPe:LENergy:LE1M
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PACKets:PTYPe:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PACKets:PTYPe:LENergy:LE2M
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → RsCmwBluetoothSig.enums.LePacketType2

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:PACKets:PTYPe:LENergy[:LE1M]
value: enums.LePacketType2 = driver.configure.connection.packets.ptype.
↪lowEnergy.get_le_1_m()
```

Specifies the type of the LE test packet. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

return packet_type: RFPHytest | RFCTe 'RFPHytest': test packet according to Bluetooth specification up to version 5.0 'RFCTe': test packet with CTE according to Bluetooth specification version 5.1

get_le_2_m() → RsCmwBluetoothSig.enums.LePacketType2

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:PACKets:PTYPe:LENergy:LE2M
value: enums.LePacketType2 = driver.configure.connection.packets.ptype.
↪lowEnergy.get_le_2_m()
```

Specifies the type of the LE test packet. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

return packet_type: RFPHytest | RFCTe 'RFPHytest': test packet according to Bluetooth specification up to version 5.0 'RFCTe': test packet with CTE according to Bluetooth specification version 5.1

get_lrange() → RsCmwBluetoothSig.enums.LePacketType2

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:PACKets:PTYPe:LENergy:LRANge
value: enums.LePacketType2 = driver.configure.connection.packets.ptype.
↪lowEnergy.get_lrange()
```

No command help available

return packet_type: No help available

set_le_1_m(*packet_type: RsCmwBluetoothSig.enums.LePacketType2*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:PACKets:PTYPe:LENergy[:LE1M]
driver.configure.connection.packets.ptype.lowEnergy.set_le_1_m(packet_type =␣
↪enums.LePacketType2.RFCTe)
```

Specifies the type of the LE test packet. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

param packet_type RFPHytest | RFCTe 'RFPHytest': test packet according to Bluetooth specification up to version 5.0 'RFCTe': test packet with CTE according to Bluetooth specification version 5.1

set_le_2_m(*packet_type: RsCmwBluetoothSig.enums.LePacketType2*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:PACKets:PTYPe:LENergy:LE2M
driver.configure.connection.packets.ptype.lowEnergy.set_le_2_m(packet_type =␣
↪enums.LePacketType2.RFCTe)
```

Specifies the type of the LE test packet. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

param packet_type RFPHytest | RFCTe 'RFPHytest': test packet according to Bluetooth specification up to version 5.0 'RFCTe': test packet with CTE according to Bluetooth specification version 5.1

set_lrange(*packet_type: RsCmwBluetoothSig.enums.LePacketType2*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:PACKets:PTYPe:LENergy:LRANge
driver.configure.connection.packets.ptype.lowEnergy.set_lrange(packet_type =␣
↪enums.LePacketType2.RFCTe)
```

No command help available

param packet_type No help available

7.1.4.2.2 PacketLength

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PACKets:PLENgth:BRATe
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PACKets:PLENgth:EDRate
```

class PacketLength

PacketLength commands group definition. 5 total commands, 1 Sub-groups, 2 group commands

get_brate() → List[int]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PACKets:PLENgth:BRATe
value: List[int] = driver.configure.connection.packets.packetLength.get_brate()
```

Sets the payload length for BR test mode.

return payload_length: numeric Range: Three values, see table below

get_edrate() → List[int]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:PACKets:PLENgth:EDRate
value: List[int] = driver.configure.connection.packets.packetLength.get_edrate()
```

Sets the payload length for EDR test mode.

return payload_length: numeric Range: Six values, see table below

set_brate(*payload_length: List[int]*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PACKets:PLENgth:BRATe
driver.configure.connection.packets.packetLength.set_brate(payload_length = [1,␣
↪2, 3])
```

Sets the payload length for BR test mode.

param payload_length numeric Range: Three values, see table below

set_edrate(*payload_length: List[int]*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:PACKets:PLENgth:EDRate
driver.configure.connection.packets.packetLength.set_edrate(payload_length = [1,
↪ 2, 3])
```

Sets the payload length for EDR test mode.

param payload_length numeric Range: Six values, see table below

Cloning the Group

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

Subgroups

7.1.4.2.2.1 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PACKets:PLENgth:LENergy:LE1M
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PACKets:PLENgth:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PACKets:PLENgth:LENergy:LE2M
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:PACKets:PLENgth:LENergy[:LE1M]
value: int = driver.configure.connection.packets.packetLength.lowEnergy.get_le_
↪1_m()
```

Specifies the payload length. INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Commands for LE direct test mode: For LE 1M PHY (...:LE1M...) , LE 2M PHY (...:LE2M...) , and LE coded PHY (...:LRANge...) .

INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Commands for LE test mode: ..:TCONnection:..

return payload_length: numeric Range: 0 byte(s) to 255 byte(s) , Unit: byte

get_le_2_m() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:PACKets:PLENgth:LENergy:LE2M
value: int = driver.configure.connection.packets.packetLength.lowEnergy.get_le_
↪2_m()
```

Specifies the payload length. INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Commands for LE direct test mode: For LE 1M PHY (...:LE1M...) , LE 2M PHY (...:LE2M...) , and LE coded PHY (...:LRANge...) .

INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Commands for LE test mode: ..:TCONnection:..

return payload_length: numeric Range: 0 byte(s) to 255 byte(s) , Unit: byte

get_lrange() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:PACKets:PLENgth:LENergy:LRANge
value: int = driver.configure.connection.packets.packetLength.lowEnergy.get_
↪lrange()
```

Specifies the payload length. INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Commands for LE direct test mode: For LE 1M PHY (...:LE1M...) , LE 2M PHY (...:LE2M...) , and LE coded PHY (...:LRANge...) .

INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Commands for LE test mode: ..:TCONnection:..

return payload_length: numeric Range: 0 byte(s) to 255 byte(s) , Unit: byte

set_le_1_m(*payload_length: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:PACKets:PLENgth:LENergy[:LE1M]
driver.configure.connection.packets.packetLength.lowEnergy.set_le_1_m(payload_
↪length = 1)
```

Specifies the payload length. INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Commands for LE direct test mode: For LE 1M PHY (...:LE1M...) , LE 2M PHY (...:LE2M...) , and LE coded PHY (...:LRANge...) .

INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Commands for LE test mode: ..:TCONnection:..

param payload_length numeric Range: 0 byte(s) to 255 byte(s) , Unit: byte

set_le_2_m(*payload_length: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:PACKets:PLENgth:LENergy:LE2M
driver.configure.connection.packets.packetLength.lowEnergy.set_le_2_m(payload_
↪length = 1)
```

Specifies the payload length. INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Commands for LE direct test mode: For LE 1M PHY (...:LE1M...) , LE 2M PHY (...:LE2M...) , and LE coded PHY (...:LRANge...) .

INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Commands for LE test mode: ..:TCONnection:..

param payload_length numeric Range: 0 byte(s) to 255 byte(s) , Unit: byte

set_lrange(*payload_length: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:PACKets:PLENgth:LENergy:LRANge
driver.configure.connection.packets.packetLength.lowEnergy.set_lrange(payload_
↪length = 1)
```

Specifies the payload length. INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Commands for LE direct test mode: For LE 1M PHY (...:LE1M...) , LE 2M PHY (...:LE2M...) , and LE coded PHY (...:LRANge...) .

INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Commands for LE test mode: ..:TCONnection:..

param payload_length numeric Range: 0 byte(s) to 255 byte(s) , Unit: byte

7.1.4.2.3 Pattern

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PACKets:PATTern:BRATe
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PACKets:PATTern:EDRate
```

class Pattern

Pattern commands group definition. 5 total commands, 1 Sub-groups, 2 group commands

get_brate() → RsCmwBluetoothSig.enums.LeRangePaternType

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PACKets:PATTern:BRATe
value: enums.LeRangePaternType = driver.configure.connection.packets.pattern.
↪get_brate()
```

Select the bit pattern to be used for tests. INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Commands for test mode classic: For BR (...:BRATe...) , EDR (...:EDRate...)
- Commands for LE direct test mode: For LE 1M PHY (...:LE1M...) , LE 2M PHY (...:LE2M...)
- Commands for LE test mode: ..:TCONnection:..

return pattern_type: ALL0 | ALL1 | P11 | P44 | PRBS9 | ALT ALL0: 00000000 ALL1: 11111111 P11: 10101010 P44: 11110000 PRBS9: pseudo-random bit sequences of a length of 9 bits (transmission of identical packet series) ALT: the periodical alternation of the pattern P11 and P44

get_edrate() → RsCmwBluetoothSig.enums.LeRangePaternType

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:PACKets:PATTern:EDRate
value: enums.LeRangePaternType = driver.configure.connection.packets.pattern.
↪get_edrate()
```

Select the bit pattern to be used for tests. INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Commands for test mode classic: For BR (...:BRATe...) , EDR (...:EDRate...)
- Commands for LE direct test mode: For LE 1M PHY (...:LE1M...) , LE 2M PHY (...:LE2M...)
- Commands for LE test mode: ..:TCONnection:..

return pattern_type: ALL0 | ALL1 | P11 | P44 | PRBS9 | ALT ALL0: 00000000 ALL1: 11111111 P11: 10101010 P44: 11110000 PRBS9: pseudo-random bit sequences of a length of 9 bits (transmission of identical packet series) ALT: the periodical alternation of the pattern P11 and P44

set_brate(*pattern_type: RsCmwBluetoothSig.enums.LeRangePaternType*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PACKets:PATTern:BRATe
driver.configure.connection.packets.pattern.set_brate(pattern_type = enums.
↪LeRangePaternType.ALL0)
```

Select the bit pattern to be used for tests. INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Commands for test mode classic: For BR (...:BRATe...) , EDR (...:EDRate...)
- Commands for LE direct test mode: For LE 1M PHY (...:LE1M...) , LE 2M PHY (...:LE2M...)
- Commands for LE test mode: ..:TCONnection:..

param pattern_type ALL0 | ALL1 | P11 | P44 | PRBS9 | ALT ALL0: 00000000 ALL1: 11111111 P11: 10101010 P44: 11110000 PRBS9: pseudo-random bit sequences of a

length of 9 bits (transmission of identical packet series) ALT: the periodical alternation of the pattern P11 and P44

set_edrate(*pattern_type: RsCmwBluetoothSig.enums.LeRangePaternType*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:PACKets:PATTern:EDRate
driver.configure.connection.packets.pattern.set_edrate(pattern_type = enums.
↪LeRangePaternType.ALL0)
```

Select the bit pattern to be used for tests. INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Commands for test mode classic: For BR (...:BRATe...) , EDR (...:EDRate...)
- Commands for LE direct test mode: For LE 1M PHY (...:LE1M...) , LE 2M PHY (...:LE2M...)
- Commands for LE test mode: ..:TCONnection:..

param pattern_type ALL0 | ALL1 | P11 | P44 | PRBS9 | ALT ALL0: 00000000 ALL1: 11111111 P11: 10101010 P44: 11110000 PRBS9: pseudo-random bit sequences of a length of 9 bits (transmission of identical packet series) ALT: the periodical alternation of the pattern P11 and P44

Cloning the Group

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

Subgroups

7.1.4.2.3.1 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PACKets:PATTern:LENergy:LE1M
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PACKets:PATTern:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PACKets:PATTern:LENergy:LE2M
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → RsCmwBluetoothSig.enums.LeRangePaternType

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:PACKets:PATTern:LENergy[:LE1M]
value: enums.LeRangePaternType = driver.configure.connection.packets.pattern.
↪lowEnergy.get_le_1_m()
```

Select the bit pattern to be used for tests. INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Commands for test mode classic: For BR (...:BRATe...) , EDR (...:EDRate...)
- Commands for LE direct test mode: For LE 1M PHY (...:LE1M...) , LE 2M PHY (...:LE2M...)
- Commands for LE test mode: ..:TCONnection:..

return pattern_type: ALL0 | ALL1 | P11 | P44 | PRBS9 | ALT ALL0: 00000000 ALL1: 11111111 P11: 10101010 P44: 11110000 PRBS9: pseudo-random bit sequences of a length of 9 bits (transmission of identical packet series) ALT: the periodical alternation of the pattern P11 and P44

`get_le_2_m()` → RsCmwBluetoothSig.enums.LeRangePaternType

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:PACKets:PATTern:LENergy:LE2M
value: enums.LeRangePaternType = driver.configure.connection.packets.pattern.
↪lowEnergy.get_le_2_m()
```

Select the bit pattern to be used for tests. INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Commands for test mode classic: For BR (...:BRATe...) , EDR (...:EDRate...)
- Commands for LE direct test mode: For LE 1M PHY (...:LE1M...) , LE 2M PHY (...:LE2M...)
- Commands for LE test mode: ..:TCONnection:..

return pattern_type: ALL0 | ALL1 | P11 | P44 | PRBS9 | ALT ALL0: 00000000 ALL1: 11111111 P11: 10101010 P44: 11110000 PRBS9: pseudo-random bit sequences of a length of 9 bits (transmission of identical packet series) ALT: the periodical alternation of the pattern P11 and P44

`get_lrange()` → RsCmwBluetoothSig.enums.LeRangePaternType

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:PACKets:PATTern:LENergy:LRANge
value: enums.LeRangePaternType = driver.configure.connection.packets.pattern.
↪lowEnergy.get_lrange()
```

Select the bit pattern to be used for tests on LE coded PHY.

return pattern_type: ALL1 | P11 | P44 | PRBS9 ALL1: 11111111 P11: 10101010 P44: 11110000 PRBS9: pseudo-random bit sequences of a length of 9 bits (transmission of identical packet series)

`set_le_1_m`(*pattern_type: RsCmwBluetoothSig.enums.LeRangePaternType*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:PACKets:PATTern:LENergy[:LE1M]
driver.configure.connection.packets.pattern.lowEnergy.set_le_1_m(pattern_type =␣
↪enums.LeRangePaternType.ALL0)
```

Select the bit pattern to be used for tests. INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Commands for test mode classic: For BR (...:BRATe...) , EDR (...:EDRate...)

- Commands for LE direct test mode: For LE 1M PHY (...:LE1M...) , LE 2M PHY (...:LE2M...)
- Commands for LE test mode: ..:TCONnection:..

param pattern_type ALL0 | ALL1 | P11 | P44 | PRBS9 | ALT ALL0: 00000000 ALL1: 11111111 P11: 10101010 P44: 11110000 PRBS9: pseudo-random bit sequences of a length of 9 bits (transmission of identical packet series) ALT: the periodical alternation of the pattern P11 and P44

set_le_2_m(*pattern_type: RsCmwBluetoothSig.enums.LeRangePaternType*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:PACKets:PATTern:LENergy:LE2M
driver.configure.connection.packets.pattern.lowEnergy.set_le_2_m(pattern_type =␣
↪enums.LeRangePaternType.ALL0)
```

Select the bit pattern to be used for tests. INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Commands for test mode classic: For BR (...:BRATe...) , EDR (...:EDRate...)
- Commands for LE direct test mode: For LE 1M PHY (...:LE1M...) , LE 2M PHY (...:LE2M...)
- Commands for LE test mode: ..:TCONnection:..

param pattern_type ALL0 | ALL1 | P11 | P44 | PRBS9 | ALT ALL0: 00000000 ALL1: 11111111 P11: 10101010 P44: 11110000 PRBS9: pseudo-random bit sequences of a length of 9 bits (transmission of identical packet series) ALT: the periodical alternation of the pattern P11 and P44

set_lrange(*pattern_type: RsCmwBluetoothSig.enums.LeRangePaternType*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:PACKets:PATTern:LENergy:LRANge
driver.configure.connection.packets.pattern.lowEnergy.set_lrange(pattern_type =␣
↪enums.LeRangePaternType.ALL0)
```

Select the bit pattern to be used for tests on LE coded PHY.

param pattern_type ALL1 | P11 | P44 | PRBS9 ALL1: 11111111 P11: 10101010 P44: 11110000 PRBS9: pseudo-random bit sequences of a length of 9 bits (transmission of identical packet series)

7.1.4.2.4 Units

class Units

Units commands group definition. 2 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.4.2.4.1 Cte

class Cte

Cte commands group definition. 2 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.4.2.4.2 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PACKets:UNITs:CTE:LENergy:LE1M
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PACKets:UNITs:CTE:LENergy:LE2M
```

class LowEnergy

LowEnergy commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

get_le_1_m() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:PACKets:UNITs:CTE:LENergy:LE1M
value: int = driver.configure.connection.packets.units.cte.lowEnergy.get_le_1_
↪m()
```

Sets the number of CTE units. One CTE unit corresponds to 8 s. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

return cte_units: integer Range: 2 to 20

get_le_2_m() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:PACKets:UNITs:CTE:LENergy:LE2M
value: int = driver.configure.connection.packets.units.cte.lowEnergy.get_le_2_
↪m()
```

Sets the number of CTE units. One CTE unit corresponds to 8 s. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

return cte_units: integer Range: 2 to 20

set_le_1_m(*cte_units: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:PACKets:UNITs:CTE:LENergy:LE1M
driver.configure.connection.packets.units.cte.lowEnergy.set_le_1_m(cte_units =␣
↪1)
```

Sets the number of CTE units. One CTE unit corresponds to 8 s. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

param cte_units integer Range: 2 to 20

set_le_2_m(*cte_units: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:PACKets:UNITs:CTE:LENergy:LE2M
driver.configure.connection.packets.units.cte.lowEnergy.set_le_2_m(cte_units =␣
↪1)
```

Sets the number of CTE units. One CTE unit corresponds to 8 s. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

param cte_units integer Range: 2 to 20

7.1.4.2.5 TypePy

class TypePy

TypePy commands group definition. 2 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.4.2.5.1 Cte

class Cte

Cte commands group definition. 2 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.4.2.5.2 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PACKets:TYPE:CTE:LENergy:LE1M
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PACKets:TYPE:CTE:LENergy:LE2M
```

class LowEnergy

LowEnergy commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

get_le_1_m() → RsCmwBluetoothSig.enums.CteType

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:PACKets:TYPE:CTE:LENergy:LE1M
value: enums.CteType = driver.configure.connection.packets.typePy.cte.lowEnergy.
↪get_le_1_m()
```

Selects the CTE type. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

return cte_type: AOA2us | AOD1us | AOD2us | AOA1us AOA1us: angle of arrival 1 µs AOA2us: angle of arrival 2 µs AOD1us: angle of departure 1 µs AOD2us: angle of departure 2 µs

get_le_2_m() → RsCmwBluetoothSig.enums.CteType

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:PACKets:TYPE:CTE:LENergy:LE2M
value: enums.CteType = driver.configure.connection.packets.typePy.cte.lowEnergy.
↪get_le_2_m()
```

Selects the CTE type. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

return cte_type: AOA2us | AOD1us | AOD2us | AOA1us AOA1us: angle of arrival 1 µs AOA2us: angle of arrival 2 µs AOD1us: angle of departure 1 µs AOD2us: angle of departure 2 µs

set_le_1_m(*cte_type: RsCmwBluetoothSig.enums.CteType*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:PACKets:TYPE:CTE:LENergy:LE1M
driver.configure.connection.packets.typePy.cte.lowEnergy.set_le_1_m(cte_type =␣
↪enums.CteType.AOA1us)
```

Selects the CTE type. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

param cte_type AOA2us | AOD1us | AOD2us | AOA1us AOA1us: angle of arrival 1 µs AOA2us: angle of arrival 2 µs AOD1us: angle of departure 1 µs AOD2us: angle of departure 2 µs

set_le_2_m(*cte_type: RsCmwBluetoothSig.enums.CteType*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:PACKets:TYPE:CTE:LENergy:LE2M
driver.configure.connection.packets.typePy.cte.lowEnergy.set_le_2_m(cte_type =␣
↪enums.CteType.AOA1us)
```

Selects the CTE type. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

param cte_type AOA2us | AOD1us | AOD2us | AOA1us AOA1us: angle of arrival 1 µs AOA2us: angle of arrival 2 µs AOD1us: angle of departure 1 µs AOD2us: angle of departure 2 µs

7.1.4.3 SynWord

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:SYNWord:LENergy
```

class SynWord

SynWord commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_low_energy() → str

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:SYNWord:LENergy
value: str = driver.configure.connection.synWord.get_low_energy()
```

Specifies the synchronization word used for LE connection.

return synch_word: hex Range: #H0 to #HFFFFFFFF

set_low_energy(*synch_word: str*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:SYNWord:LENergy
driver.configure.connection.synWord.set_low_energy(synch_word = r1)
```

Specifies the synchronization word used for LE connection.

param synch_word hex Range: #H0 to #HFFFFFFFF

7.1.4.4 Cscheme

class Cscheme
Cscheme commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.4.4.1 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:CSCHeme:LENergy:LRANge
```

class LowEnergy
LowEnergy commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_lrange() → RsCmwBluetoothSig.enums.CodingScheme

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:CSCHeme:LENergy:LRANge
value: enums.CodingScheme = driver.configure.connection.cscheme.lowEnergy.get_
↪lrange()
```

No command help available

return coding_scheme: No help available

set_lrange(*coding_scheme: RsCmwBluetoothSig.enums.CodingScheme*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:CSCHeme:LENergy:LRANge
driver.configure.connection.cscheme.lowEnergy.set_lrange(coding_scheme = enums.
↪CodingScheme.S2)
```

No command help available

param coding_scheme No help available

7.1.4.5 Fec

class Fec
Fec commands group definition. 2 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.4.5.1 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:FEC:LENergy:LRANge
```

class LowEnergy
LowEnergy commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_lrange() → RsCmwBluetoothSig.enums.CodingScheme

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:FEC:LENergy:LRANge
value: enums.CodingScheme = driver.configure.connection.fec.lowEnergy.get_
↪lrange()
```

Defines the coding S for LE coded PHY according to the core specification version 5.0 for Bluetooth wireless technolo
INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Command for LE direct test mode: ..:FEC:LENergy..

INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Command for LE connection tests (normal mode) : ..:FEC:NMODe:LENergy..

INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Command for LE test mode: ..:TCONnection:FEC:LENergy..

return coding_scheme: S8 | S2 Coding S = 8 or S = 2

set_lrange(*coding_scheme: RsCmwBluetoothSig.enums.CodingScheme*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:FEC:LENergy:LRANge
driver.configure.connection.fec.lowEnergy.set_lrange(coding_scheme = enums.
↪CodingScheme.S2)
```

Defines the coding S for LE coded PHY according to the core specification version 5.0 for Bluetooth wireless technolo
INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Command for LE direct test mode: ..:FEC:LENergy..

INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Command for LE connection tests (normal mode) : ..:FEC:NMODe:LENergy..

INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Command for LE test mode: ..:TCONnection:FEC:LENergy..

param coding_scheme S8 | S2 Coding S = 8 or S = 2

7.1.4.5.2 Nmode

class Nmode

Nmode commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.4.5.2.1 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:FEC:NMODe:LENergy:LRANge
```

class LowEnergy

LowEnergy commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_lrange() → RsCmwBluetoothSig.enums.CodingScheme

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:FEC:NMODe:LENergy:LRANge
value: enums.CodingScheme = driver.configure.connection.fec.nmode.lowEnergy.get_
↪lrange()
```

Defines the coding S for LE coded PHY according to the core specification version 5.0 for Bluetooth wireless technolo

INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Command for LE direct test mode: ..:FEC:LENergy..

INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Command for LE connection tests (normal mode) : ..:FEC:NMODe:LENergy..

INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Command for LE test mode: ..:TCONnection:FEC:LENergy..

return coding_scheme: S8 | S2 Coding S = 8 or S = 2

set_lrange(*coding_scheme: RsCmwBluetoothSig.enums.CodingScheme*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:FEC:NMODe:LENergy:LRANge
driver.configure.connection.fec.nmode.lowEnergy.set_lrange(coding_scheme =␣
↪enums.CodingScheme.S2)
```

Defines the coding S for LE coded PHY according to the core specification version 5.0 for Bluetooth wireless technolo

INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Command for LE direct test mode: ..:FEC:LENergy..

INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Command for LE connection tests (normal mode) : ..:FEC:NMODe:LENergy..

INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Command for LE test mode: ..:TCONnection:FEC:LENergy..

param coding_scheme S8 | S2 Coding S = 8 or S = 2

7.1.4.6 Phy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PHY:LENergy
```

class Phy

Phy commands group definition. 2 total commands, 1 Sub-groups, 1 group commands

get_low_energy() → RsCmwBluetoothSig.enums.LePhysicalType

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PHY:LENergy
value: enums.LePhysicalType = driver.configure.connection.phy.get_low_energy()
```

Selects the physical layer used for LE connections. INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Command for LE direct test mode: ..:PHY:LENergy

INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Command for LE connection tests (normal mode) : ..:PHY:NMODe:LENergy

INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Command for LE test mode: ..:TCONnection:PHY:LENergy

return phy: LE1M | LE2M | LELR LE1M: LE 1 Msymbol/s uncoded PHY LE2M: LE 2 Msymbol/s uncoded PHY LELR: LE 1 Msymbol/s long range (LE coded PHY)

set_low_energy(*phy: RsCmwBluetoothSig.enums.LePhysicalType*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PHY:LENergy
driver.configure.connection.phy.set_low_energy(phy = enums.LePhysicalType.LE1M)
```

Selects the physical layer used for LE connections. INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Command for LE direct test mode: ..:PHY:LENergy

INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Command for LE connection tests (normal mode) : ..:PHY:NMODe:LENergy

INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Command for LE test mode: ..:TCONnection:PHY:LENergy

param phy LE1M | LE2M | LELR LE1M: LE 1 Msymbol/s uncoded PHY LE2M: LE 2 Msymbol/s uncoded PHY LELR: LE 1 Msymbol/s long range (LE coded PHY)

Cloning the Group

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

Subgroups

7.1.4.6.1 Nmode

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PHY:NMODe:LENergy
```

class Nmode

Nmode commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_low_energy() → RsCmwBluetoothSig.enums.LePhysicalType

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PHY:NMODe:LENergy
value: enums.LePhysicalType = driver.configure.connection.phy.nmode.get_low_
↪energy()
```

Selects the physical layer used for LE connections. INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Command for LE direct test mode: ..:PHY:LENergy

INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Command for LE connection tests (normal mode) : ..:PHY:NMODe:LENergy

INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Command for LE test mode: ..:TCONnection:PHY:LENergy

return phy: LE1M | LE2M | LELR LE1M: LE 1 Msymbol/s uncoded PHY LE2M: LE 2 Msymbol/s uncoded PHY LELR: LE 1 Msymbol/s long range (LE coded PHY)

set_low_energy(*phy: RsCmwBluetoothSig.enums.LePhysicalType*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PHY:NMODe:LENergy
driver.configure.connection.phy.nmode.set_low_energy(phy = enums.LePhysicalType.
↪LE1M)
```

Selects the physical layer used for LE connections. INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Command for LE direct test mode: ..:PHY:LENergy

INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Command for LE connection tests (normal mode) : ..:PHY:NMODe:LENergy

INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Command for LE test mode: ..:TCONnection:PHY:LENergy

param phy LE1M | LE2M | LELR LE1M: LE 1 Msymbol/s uncoded PHY LE2M: LE 2 Msymbol/s uncoded PHY LELR: LE 1 Msymbol/s long range (LE coded PHY)

7.1.4.7 PowerControl

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PCONtrol:EPCMode
```

class PowerControl

PowerControl commands group definition. 5 total commands, 3 Sub-groups, 1 group commands

get_epc_mode() → RsCmwBluetoothSig.enums.PowerControlMode

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PCONtrol:EPCMode
value: enums.PowerControlMode = driver.configure.connection.powerControl.get_
↪epc_mode()
```

Activates/deactivates enhanced power control mode.

return pc_mode: AUTO | OFF AUTO : instrument uses enhanced power control if EUT supports it, otherwise it uses legacy power control OFF : instrument uses legacy power control

set_epc_mode(*pc_mode: RsCmwBluetoothSig.enums.PowerControlMode*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PCONtrol:EPCMode
driver.configure.connection.powerControl.set_epc_mode(pc_mode = enums.
↪PowerControlMode.AUTO)
```

Activates/deactivates enhanced power control mode.

param pc_mode AUTO | OFF AUTO : instrument uses enhanced power control if EUT supports it, otherwise it uses legacy power control OFF : instrument uses legacy power control

Cloning the Group

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

Subgroups

7.1.4.7.1 Step

class Step

Step commands group definition. 2 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.4.7.1.1 Action

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PCONtrol:STEP:ACTion:LESignaling
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PCONtrol:STEP:ACTion
```

class Action

Action commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

set_le_signaling(*pcontrol: RsCmwBluetoothSig.enums.PowerControl*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:PCONtrol:STEP:ACTion:LESignaling
driver.configure.connection.powerControl.step.action.set_le_signaling(pcontrol␣
↪= enums.PowerControl.DOWN)
```

Sends a command to the DUT to increase/decrease power.

param pcontrol UP | DOWN | MAX One step up, one step down, command to maximum DUT Tx power

set_value(*pcontrol: RsCmwBluetoothSig.enums.PowerControl*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PCONtrol:STEP:ACTion
driver.configure.connection.powerControl.step.action.set_value(pcontrol = enums.
↪PowerControl.DOWN)
```

Sends a command to the EUT to increase/decrease power.

param pcontrol UP | DOWN | MAX One step up, one step down, command to maximum EUT power

7.1.4.7.2 Ssize

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PCONtrol:SSIZe:LESignaling
```

class Ssize

Ssize commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_le_signaling() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:PCONtrol:SSIZe:LESignaling
value: int = driver.configure.connection.powerControl.ssize.get_le_signaling()
```

Sets the step size for increasing / decreasing the Tx power.

return stepsize: numeric Range: 0 dB to 10 dB

set_le_signaling(*stepsize: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:PCONtrol:SSIZe:LESignaling
driver.configure.connection.powerControl.ssize.set_le_signaling(stepsize = 1)
```

Sets the step size for increasing / decreasing the Tx power.

param stepsize numeric Range: 0 dB to 10 dB

7.1.4.7.3 PcMode

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PCONtrol:PCMode:LESignaling
```

class PcMode

PcMode commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_le_signaling() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:PCONtrol:PCMode:LESignaling
value: bool = driver.configure.connection.powerControl.pcMode.get_le_signaling()
```

Specifies, whether the power commands respect the reported EUT capabilities or ignore them.

return override_capabilites: OFF | ON OFF: The reported EUT capabilities are respected. ON: The reported EUT capabilities are ignored.

set_le_signaling(*override_capabilites: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:PCONtrol:PCMode:LESignaling
driver.configure.connection.powerControl.pcMode.set_le_signaling(override_
↪capabilites = False)
```

Specifies, whether the power commands respect the reported EUT capabilities or ignore them.

param override_capabilites OFF | ON OFF: The reported EUT capabilities are respected. ON: The reported EUT capabilities are ignored.

7.1.4.8 Paging

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PAGing:PSRMode
```

class Paging

Paging commands group definition. 5 total commands, 2 Sub-groups, 1 group commands

get_psr_mode() → RsCmwBluetoothSig.enums.PsrMode

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PAGing:PSRMode
value: enums.PsrMode = driver.configure.connection.paging.get_psr_mode()
```

Sets/gets the page scan repetition mode to be used for the default device (see method RsCmwBluetoothSig.Configure. Connection.BdAddress.eut) .

return psr_mode: R0 | R1 | R2 Paging mode R0, R1, R2. Select the value according to the page scan repetition mode of the default device.

set_psr_mode(*psr_mode: RsCmwBluetoothSig.enums.PsrMode*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PAGing:PSRMode
driver.configure.connection.paging.set_psr_mode(psr_mode = enums.PsrMode.R0)
```

Sets/gets the page scan repetition mode to be used for the default device (see method RsCmwBluetoothSig.Configure. Connection.BdAddress.eut) .

param psr_mode R0 | R1 | R2 Paging mode R0, R1, R2. Select the value according to the page scan repetition mode of the default device.

Cloning the Group

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

Subgroups

7.1.4.8.1 Timeout

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PAGing:TOUT:LESignaling
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PAGing:TOUT
```

class Timeout

Timeout commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

get_le_signaling() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:PAGing:TOUT:LESignaling
value: int = driver.configure.connection.paging.timeout.get_le_signaling()
```

Specifies the PageTimeout configuration parameter, i.e. the maximum time the local link manager waits for a baseband page response from the EUT.

return timeout: integer Range: 10 ms to 30E+3 ms, Unit: s

get_value() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PAGing:TOUT
value: int = driver.configure.connection.paging.timeout.get_value()
```

Sets/gets the Page_Timeout configuration parameter, i.e. the maximum time the local link manager waits for a baseband page response from the EUT.

return timeout: integer Range: 22 to 65535, Unit: slot (625 µs)

set_le_signaling(*timeout: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:PAGing:TOUT:LESignaling
driver.configure.connection.paging.timeout.set_le_signaling(timeout = 1)
```

Specifies the PageTimeout configuration parameter, i.e. the maximum time the local link manager waits for a baseband page response from the EUT.

param timeout integer Range: 10 ms to 30E+3 ms, Unit: s

set_value(*timeout: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PAGing:TOUT
driver.configure.connection.paging.timeout.set_value(timeout = 1)
```

Sets/gets the Page_Timeout configuration parameter, i.e. the maximum time the local link manager waits for a baseband page response from the EUT.

param timeout integer Range: 22 to 65535, Unit: slot (625 µs)

7.1.4.8.2 Ptarget

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PAGing:PTARget:LESignaling
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PAGing:PTARget
```

class Ptarget

Ptarget commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

get_le_signaling() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:PAGing:PTARget:LESignaling
value: int = driver.configure.connection.paging.ptarget.get_le_signaling()
```

Selects the EUT for paging. The default device is 0. For the inquiry results, see method RsCmwBluetoothSig.Configure. Connection.Inquiry.Ptargets.Catalog.leSignaling.

return target: numeric Sequence number of device listed in inquiry results.

get_value() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PAGing:PTARget
value: int = driver.configure.connection.paging.ptarget.get_value()
```

Selects the device to page from the paging target catalog (see method RsCmwBluetoothSig.Configure.Connection.Inquiry. Ptargets.Catalog.value) . After a reset, if no inquiry was made before or if no device was detected during the previous inquiry, only the default device (<Target>=0) can be selected. After a successful inquiry, the first discovered device (<Target>=1) is pre-selected.

return target: numeric Index of the device in the paging target catalog, where 0 always corresponds to the default device. If an invalid index is selected, an error message is returned. Range: Integer = 0

set_le_signaling(*target: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:PAGing:PTARget:LESignaling
driver.configure.connection.paging.ptarget.set_le_signaling(target = 1)
```

Selects the EUT for paging. The default device is 0. For the inquiry results, see method RsCmwBluetoothSig.Configure. Connection.Inquiry.Ptargets.Catalog.leSignaling.

param target numeric Sequence number of device listed in inquiry results.

set_value(*target: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PAGing:PTARget
driver.configure.connection.paging.ptarget.set_value(target = 1)
```

Selects the device to page from the paging target catalog (see method RsCmwBluetoothSig.Configure.Connection.Inquiry. Ptargets.Catalog.value) . After a reset, if no inquiry was made before or if no device was detected during the previous inquiry, only the default device (<Target>=0) can be selected. After a successful inquiry, the first discovered device (<Target>=1) is pre-selected.

param target numeric Index of the device in the paging target catalog, where 0 always corresponds to the default device. If an invalid index is selected, an error message is returned. Range: Integer = 0

7.1.4.9 BdAddress

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:BDADdress:CMW
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:BDADdress:EUT
```

class BdAddress

BdAddress commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

get_cmw() → str

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:BDADdress:CMW
value: str = driver.configure.connection.bdAddress.get_cmw()
```

Sets/gets the Bluetooth device address (BD_ADDR) of the R&S CMW.

return bd_address: hex Range: #H0 to #HFFFFFFFFFFFF (12 hexadecimal digits)

get_eut() → str

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:BDADdress:EUT
value: str = driver.configure.connection.bdAddress.get_eut()
```

Sets/gets the Bluetooth device address (BD_ADDR) of a default device to attempt a connection to. If no inquiry was made before, this BD_ADDR is used for paging; otherwise, the device to page can be set via method RsCmwBluetoothSig.Configure. Connection.Paging.Ptarget.value.

return bd_address: hex Range: #H0 to #HFFFFFFFFFFFF (12 hexadecimal digits)

set_cmw(*bd_address: str*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:BDADdress:CMW
driver.configure.connection.bdAddress.set_cmw(bd_address = r1)
```

Sets/gets the Bluetooth device address (BD_ADDR) of the R&S CMW.

param bd_address hex Range: #H0 to #HFFFFFFFFFFFF (12 hexadecimal digits)

set_eut(*bd_address: str*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:BDADdress:EUT
driver.configure.connection.bdAddress.set_eut(bd_address = r1)
```

Sets/gets the Bluetooth device address (BD_ADDR) of a default device to attempt a connection to. If no inquiry was made before, this BD_ADDR is used for paging; otherwise, the device to page can be set via method RsCmwBluetoothSig.Configure. Connection.Paging.Ptarget.value.

param bd_address hex Range: #H0 to #HFFFFFFFFFFFF (12 hexadecimal digits)

7.1.4.10 Inquiry

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:INQuiry:ILENgth
```

class Inquiry

Inquiry commands group definition. 8 total commands, 5 Sub-groups, 1 group commands

get_ilength() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:INQuiry:ILENgth
value: int = driver.configure.connection.inquiry.get_ilength()
```

Sets/gets the Inquiry_Length parameter, i.e. the total duration of the inquiry mode.

return inquiry_length: numeric The inquiry length in units of 1.28 s Range: 1 to 24, Unit: 1.28 s

set_ilength(*inquiry_length: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:INQuiry:ILENgth
driver.configure.connection.inquiry.set_ilength(inquiry_length = 1)
```

Sets/gets the Inquiry_Length parameter, i.e. the total duration of the inquiry mode.

param inquiry_length numeric The inquiry length in units of 1.28 s Range: 1 to 24, Unit: 1.28 s

Cloning the Group

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

Subgroups

7.1.4.10.1 Ptargets

class Ptargets
Ptargets commands group definition. 2 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.4.10.1.1 Catalog

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:INQuiry:PTARgets:CATalog:LESignaling
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:INQuiry:PTARgets:CATalog
```

class Catalog
Catalog commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

class LeSignalingStruct
Structure for reading output parameters. Fields:

- No_Discovered_Devices: int: No parameter help available
- Item_Number: List[int]: No parameter help available
- Discovered_Eut: List[str]: string A comma-separated list of Bluetooth devices, where each device is represented by an item number and its Address in hexadecimal notation. Item number 0 always represents the default target.

class ValueStruct
Structure for reading output parameters. Fields:

- No_Discovered_Devices: int: No parameter help available
- Item_Number: List[int]: No parameter help available
- Discovered_Eut: List[str]: string A comma-separated list of Bluetooth devices, where each device is represented by an item number and its BD_Address in hexadecimal notation. Item number 0 always represents the default target.

get_le_signaling() → LeSignalingStruct

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:INQuiry:PTARgets:CATalog:LESignaling
value: LeSignalingStruct = driver.configure.connection.inquiry.ptargets.catalog.
↪get_le_signaling()
```

This query returns a list of all targets available for paging, i.e. all LE devices found. If no inquiry was made before, this list only contains the default device (see method RsCmwBluetoothSig.Configure.Connection.Address.Eut.leSignaling) . After inquiry, it also contains the devices that were responding (in chronological order) .

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

get_value() → ValueStruct

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:INQuiry:PTARgets:CATalog
value: ValueStruct = driver.configure.connection.inquiry.ptargets.catalog.get_
↪value()
```

This query returns a list of all targets available for paging. If no inquiry was made before, this list only contains the default device (see method RsCmwBluetoothSig.Configure.Connection.BdAddress.eut) . After inquiry, it also contains the devices that were responding (in chronological order) .

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

7.1.4.10.2 NoResponses

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:INQuiry:NOResponses:LESignaling
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:INQuiry:NOResponses
```

class NoResponses

NoResponses commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

get_le_signaling() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:INQuiry:NOResponses:LESignaling
value: int = driver.configure.connection.inquiry.noResponses.get_le_signaling()
```

Specifies the maximum number of responses recorded during an inquiry.

return number_responses: numeric Range: 1 to 100

get_value() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:INQuiry:NOResponses
value: int = driver.configure.connection.inquiry.noResponses.get_value()
```

Sets/gets the maximum number of responses recorded during an inquiry.

return number_responses: numeric The maximum number of responses, where 0 means 'unlimited'. Range: 0 to 12

set_le_signaling(*number_responses: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:INQuiry:NOResponses:LESignaling
driver.configure.connection.inquiry.noResponses.set_le_signaling(number_
↪responses = 1)
```

Specifies the maximum number of responses recorded during an inquiry.

param number_responses numeric Range: 1 to 100

set_value(*number_responses: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:INQuiry:NOResponses
driver.configure.connection.inquiry.noResponses.set_value(number_responses = 1)
```

Sets/gets the maximum number of responses recorded during an inquiry.

param number_responses numeric The maximum number of responses, where 0 means 'unlimited'. Range: 0 to 12

7.1.4.10.3 Sinterval

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:INQuiry:SINTerval:LESignaling
```

class Sinterval

Sinterval commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_le_signaling() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:INQuiry:SINTerval:LESignaling
value: int = driver.configure.connection.inquiry.sinterval.get_le_signaling()
```

Specifies the Inquiry Scan Interval between two consecutive inquiry scans. The interval in ms is calculated as the specified value multiplied by 0.625 ms.

return inq_scan_int: numeric Range: 4 to 16.384E+3

set_le_signaling(*inq_scan_int: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:INQuiry:SINTerval:LESignaling
driver.configure.connection.inquiry.sinterval.set_le_signaling(inq_scan_int = 1)
```

Specifies the Inquiry Scan Interval between two consecutive inquiry scans. The interval in ms is calculated as the specified value multiplied by 0.625 ms.

param inq_scan_int numeric Range: 4 to 16.384E+3

7.1.4.10.4 Duration

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:INQuiry:DURation:LESignaling
```

class Duration

Duration commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_le_signaling() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:INQuiry:DURation:LESignaling
value: int = driver.configure.connection.inquiry.duration.get_le_signaling()
```

Specifies the total inquiry duration.

return inq_duration: numeric Range: 5 ms to 30E+3 ms

set_le_signaling(*inq_duration: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:INQuiry:DURation:LESignaling
driver.configure.connection.inquiry.duration.set_le_signaling(inq_duration = 1)
```

Specifies the total inquiry duration.

param inq_duration numeric Range: 5 ms to 30E+3 ms

7.1.4.10.5 Swindow

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:INQuiry:SWINdow:LESignaling
```

class Swindow

Swindow commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_le_signaling() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:INQuiry:SWINdow:LESignaling
value: int = driver.configure.connection.inquiry.swindow.get_le_signaling()
```

Specifies Inquiry Scan Window - the inquiry scan duration. The duration in ms is calculated as the specified value multiplied by 0.625 ms.

return inq_scan_window: numeric Range: 4 to 16.384E+3

set_le_signaling(*inq_scan_window: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:INQuiry:SWINdow:LESignaling
driver.configure.connection.inquiry.swindow.set_le_signaling(inq_scan_window =␣
↪1)
```

Specifies Inquiry Scan Window - the inquiry scan duration. The duration in ms is calculated as the specified value multiplied by 0.625 ms.

param inq_scan_window numeric Range: 4 to 16.384E+3

7.1.4.11 EutCharacter

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:EUTCharacter:OPCMode
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:EUTCharacter:SNBehaviour
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:EUTCharacter:TCPChange
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:EUTCharacter:RLSettling
```

class EutCharacter

EutCharacter commands group definition. 4 total commands, 0 Sub-groups, 4 group commands

get_opc_mode() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:EUTCharacter:OPCMode
value: bool = driver.configure.connection.eutCharacter.get_opc_mode()
```

No command help available

return opc_mode: No help available

get_rl_settling() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:EUTCharacter:RLSettling
value: float = driver.configure.connection.eutCharacter.get_rl_settling()
```

No command help available

return settling_time: No help available

get_sn_behaviour() → RsCmwBluetoothSig.enums.SequenceNumbering

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:EUTCharacter:SNBehaviour
value: enums.SequenceNumbering = driver.configure.connection.eutCharacter.get_
↪sn_behaviour()
```

No command help available

return seq_numbering: No help available

get_tcp_change() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:EUTCharacter:TCPChange
value: bool = driver.configure.connection.eutCharacter.get_tcp_change()
```

No command help available

return tcp_change: No help available

set_opc_mode(*opc_mode: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:EUTCharacter:OPCMode
driver.configure.connection.eutCharacter.set_opc_mode(opc_mode = False)
```

No command help available

param opc_mode No help available

set_rl_settling(*settling_time: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:EUTCharacter:RLSettling
driver.configure.connection.eutCharacter.set_rl_settling(settling_time = 1.0)
```

No command help available

param settling_time No help available

set_sn_behaviour(*seq_numbering: RsCmwBluetoothSig.enums.SequenceNumbering*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:EUTCharacter:SNBehaviour
driver.configure.connection.eutCharacter.set_sn_behaviour(seq_numbering = enums.
↪SequenceNumbering.NORM)
```

No command help available

param seq_numbering No help available

set_tcp_change(*tcp_change: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:EUTCharacter:TCPChange
driver.configure.connection.eutCharacter.set_tcp_change(tcp_change = False)
```

No command help available

param tcp_change No help available

7.1.4.12 WfcMap

class WfcMap
WfcMap commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.4.12.1 LeSignaling

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:WFCMap:LESignaling:CCENtral
```

class LeSignaling
LeSignaling commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_ccentral() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:WFCMap:LESignaling:CCENtral
value: int = driver.configure.connection.wfcMap.leSignaling.get_ccentral()
```

Specifies the number of connection events to wait before checking the channel map change.

return wait_for_ch_map: numeric Range: 6 to 100

set_ccentral(*wait_for_ch_map: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:WFCMap:LESignaling:CCENtral
driver.configure.connection.wfcMap.leSignaling.set_ccentral(wait_for_ch_map = 1)
```

Specifies the number of connection events to wait before checking the channel map change.

param wait_for_ch_map numeric Range: 6 to 100

7.1.4.13 Slatency

class Slatency
Slatency commands group definition. 2 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.4.13.1 LeSignaling

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:SLATency:LESignaling:CPERipheral
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:SLATency:LESignaling:CCENtral
```

class LeSignaling

LeSignaling commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

get_ccentral() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:SLATency:LESignaling[:CCENtral]
value: int = driver.configure.connection.slatency.leSignaling.get_ccentral()
```

Specify the latency of slave responses for connection tests, central ..:CCENtral and peripheral ..:CPERipheralR&S CMW LE role.

return slave_latency: numeric Range: 0 to 499, Unit: The number of consecutive connection events

get_cperipheral() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:SLATency:LESignaling:CPERipheral
value: int = driver.configure.connection.slatency.leSignaling.get_cperipheral()
```

Specify the latency of slave responses for connection tests, central ..:CCENtral and peripheral ..:CPERipheralR&S CMW LE role.

return slave_latency: numeric Range: 0 to 499, Unit: The number of consecutive connection events

set_ccentral(*slave_latency: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:SLATency:LESignaling[:CCENtral]
driver.configure.connection.slatency.leSignaling.set_ccentral(slave_latency = 1)
```

Specify the latency of slave responses for connection tests, central ..:CCENtral and peripheral ..:CPERipheralR&S CMW LE role.

param slave_latency numeric Range: 0 to 499, Unit: The number of consecutive connection events

set_cperipheral(*slave_latency: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:SLATency:LESignaling:CPERipheral
driver.configure.connection.slatency.leSignaling.set_cperipheral(slave_latency␣
↪= 1)
```

Specify the latency of slave responses for connection tests, central ..:CCENtral and peripheral ..:CPERipheralR&S CMW LE role.

param slave_latency numeric Range: 0 to 499, Unit: The number of consecutive connection events

7.1.4.14 Rencryption

class Rencryption

Rencryption commands group definition. 2 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.4.14.1 LeSignaling

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:RENCryption:LESignaling:CCENtral
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:RENCryption:LESignaling:CPERipheral
```

class LeSignaling

LeSignaling commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

get_ccentral() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:RENCryption:LESignaling:CCENtral
value: bool = driver.configure.connection.rencryption.leSignaling.get_ccentral()
```

Specifies, whether the encryption request from the EUT is accepted or rejected by the R&S CMW in central (..:CCENtral) or peripheral (..:CPERipheral) LE role. The command is relevant for LE connection tests.

return rej_encryption: OFF | ON OFF: accept encryption request ON: reject encryption request

get_cperipheral() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:RENCryption:LESignaling:CPERipheral
value: bool = driver.configure.connection.rencryption.leSignaling.get_
↪cperipheral()
```

Specifies, whether the encryption request from the EUT is accepted or rejected by the R&S CMW in central (..:CCENtral) or peripheral (..:CPERipheral) LE role. The command is relevant for LE connection tests.

return rej_encryption: OFF | ON OFF: accept encryption request ON: reject encryption request

set_ccentral(*rej_encryption: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:RENCryption:LESignaling:CCENtral
driver.configure.connection.rencryption.leSignaling.set_ccentral(rej_encryption␣
↪= False)
```

Specifies, whether the encryption request from the EUT is accepted or rejected by the R&S CMW in central (..:CCENtral) or peripheral (..:CPERipheral) LE role. The command is relevant for LE connection tests.

param rej_encryption OFF | ON OFF: accept encryption request ON: reject encryption request

set_cperipheral(*rej_encryption: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:RENCryption:LESignaling:CPERipheral
driver.configure.connection.rencryption.leSignaling.set_cperipheral(rej_
↪encryption = False)
```

Specifies, whether the encryption request from the EUT is accepted or rejected by the R&S CMW in central (..:CCENtral) or peripheral (..:CPERipheral) LE role. The command is relevant for LE connection tests.

param rej_encryption OFF | ON OFF: accept encryption request ON: reject encryption request

7.1.4.15 Iencryption

class Iencryption

Iencryption commands group definition. 2 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.4.15.1 LeSignaling

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:IENCryption:LESignaling:CCENtral
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:IENCryption:LESignaling:CPERipheral
```

class LeSignaling

LeSignaling commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

get_ccentral() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:IENCryption:LESignaling:CCENtral
value: bool = driver.configure.connection.iencryption.leSignaling.get_ccentral()
```

Indicates the R&S CMW support of encryption in central (..:CCENtral) or peripheral (..:CPERipheral) LE role. The command is relevant for LE connection tests.

return ind_encryption: OFF | ON

get_cperipheral() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:IENCryption:LESignaling:CPERipheral
value: bool = driver.configure.connection.iencryption.leSignaling.get_
↪cperipheral()
```

Indicates the R&S CMW support of encryption in central (..:CCENtral) or peripheral (..:CPERipheral) LE role. The command is relevant for LE connection tests.

return ind_encryption: OFF | ON

set_ccentral(*ind_encryption: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:IENCryption:LESignaling:CCENtral
driver.configure.connection.iencryption.leSignaling.set_ccentral(ind_encryption␣
↪= False)
```

Indicates the R&S CMW support of encryption in central (..:CCENtral) or peripheral (..:CPERipheral) LE role. The command is relevant for LE connection tests.

param ind_encryption OFF | ON

set_cperipheral(*ind_encryption: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:IENCryption:LESignaling:CPERipheral
driver.configure.connection.iencryption.leSignaling.set_cperipheral(ind_
↪encryption = False)
```

Indicates the R&S CMW support of encryption in central (..:CCENtral) or peripheral (..:CPERipheral) LE role. The command is relevant for LE connection tests.

param ind_encryption OFF | ON

7.1.4.16 Cmw

class Cmw

Cmw commands group definition. 2 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.4.16.1 Role

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:CMW:ROLE:LESignaling
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:CMW:ROLE
```

class Role

Role commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

get_le_signaling() → RsCmwBluetoothSig.enums.SignalingCmwRole

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:CMW:ROLE:LESignaling
value: enums.SignalingCmwRole = driver.configure.connection.cmw.role.get_le_
↪signaling()
```

Sets the LE role of the instrument for LE connection tests.

return sig_cmw_role: CENTral | PERipheral

get_value() → RsCmwBluetoothSig.enums.PriorityRole

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:CMW:ROLE
value: enums.PriorityRole = driver.configure.connection.cmw.role.get_value()
```

Specifies the connection control role of the R&S CMW for audio connections.

return role: No help available

set_le_signaling(*sig_cmw_role: RsCmwBluetoothSig.enums.SignalingCmwRole*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:CMW:ROLE:LESignaling
driver.configure.connection.cmw.role.set_le_signaling(sig_cmw_role = enums.
↪SignalingCmwRole.CENTral)
```

Sets the LE role of the instrument for LE connection tests.

param sig_cmw_role CENTral | PERipheral

set_value(*role: RsCmwBluetoothSig.enums.PriorityRole*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:CMW:ROLE
driver.configure.connection.cmw.role.set_value(role = enums.PriorityRole.MASTer)
```

Specifies the connection control role of the R&S CMW for audio connections.

param role MASTer | SLAVe

7.1.4.17 Address

class Address

Address commands group definition. 3 total commands, 3 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.4.17.1 Cmw

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:ADDRess:CMW:LESignaling
```

class Cmw

Cmw commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_le_signaling() → str

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:ADDRess:CMW:LESignaling
value: str = driver.configure.connection.address.cmw.get_le_signaling()
```

Sets the public address of R&S CMW

return cmw_address: hex Range: #H0 to #HFFFFFFFFFFFF

set_le_signaling(*cmw_address: str*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:ADDRess:CMW:LESignaling
driver.configure.connection.address.cmw.set_le_signaling(cmw_address = r1)
```

Sets the public address of R&S CMW

param cmw_address hex Range: #H0 to #HFFFFFFFFFFFF

7.1.4.17.2 Eut

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:ADDRess:EUT:LESignaling
```

class Eut

Eut commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_le_signaling() → str

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:ADDRess:EUT:LESignaling
value: str = driver.configure.connection.address.eut.get_le_signaling()
```

Specifies the default EUT.

return address_def: hex Range: #H0 to #HFFFFFFFFFFFF

set_le_signaling(*address_def: str*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:ADDRess:EUT:LESignaling
driver.configure.connection.address.eut.set_le_signaling(address_def = r1)
```

Specifies the default EUT.

param address_def hex Range: #H0 to #HFFFFFFFFFFFF

7.1.4.17.3 TypePy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:ADDRess:TYPE:LESignaling
```

class TypePy

TypePy commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_le_signaling() → RsCmwBluetoothSig.enums.AddressType

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:ADDRess:TYPE:LESignaling
value: enums.AddressType = driver.configure.connection.address.typePy.get_le_
↪signaling()
```

Selects public or random addressing.

return addr_type: PUBLic | RANDom

set_le_signaling(*addr_type: RsCmwBluetoothSig.enums.AddressType*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:ADDRess:TYPE:LESignaling
driver.configure.connection.address.typePy.set_le_signaling(addr_type = enums.
↪AddressType.PUBLic)
```

Selects public or random addressing.

param addr_type PUBLic | RANDom

7.1.4.18 Raddress

class Raddress

Raddress commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.4.18.1 Cmw

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:RADDress:CMW:LESignaling
```

class Cmw

Cmw commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_le_signaling() → str

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:CONNection:RADDress:CMW:LESignaling
value: str = driver.configure.connection.raddress.cmw.get_le_signaling()
```

Queries the randomly generated address of R&S CMW

return cmw_address: hex Range: #H0 to #HFFFFFFFFFFFF

7.1.4.19 SvTimeout

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:SVTimeout:LESignaling
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:SVTimeout
```

class SvTimeout

SvTimeout commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

get_le_signaling() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:SVTimeout:LESignaling
value: int = driver.configure.connection.svTimeout.get_le_signaling()
```

Specifies the duration of tolerated connection breaks down.

return supervision_timeout: numeric Range: 100 ms to 32E+3 ms, Unit: ms

get_value() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:SVTimeout
value: int = driver.configure.connection.svTimeout.get_value()
```

Sets/gets the LMP supervision timeout.

return supervision_timeout: numeric Range: 400 slots to 65535 slots, Unit: slots

set_le_signaling(*supervision_timeout: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:SVTimeout:LESignaling
driver.configure.connection.svTimeout.set_le_signaling(supervision_timeout = 1)
```

Specifies the duration of tolerated connection breaks down.

param supervision_timeout numeric Range: 100 ms to 32E+3 ms, Unit: ms

set_value(*supervision_timeout: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:SVTimeout
driver.configure.connection.svTimeout.set_value(supervision_timeout = 1)
```

Sets/gets the LMP supervision timeout.

param supervision_timeout numeric Range: 400 slots to 65535 slots, Unit: slots

7.1.4.20 Interval

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:INTerval:LESignaling
```

class Interval

Interval commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_le_signaling() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:INTerval:LESignaling
value: int = driver.configure.connection.interval.get_le_signaling()
```

Sets the time interval between the two consecutive connection events. The interval in ms is calculated as the specified value multiplied by 1.25 ms.

return connection_interval: numeric Range: 6 to 3200

set_le_signaling(*connection_interval: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:INTerval:LESignaling
driver.configure.connection.interval.set_le_signaling(connection_interval = 1)
```

Sets the time interval between the two consecutive connection events. The interval in ms is calculated as the specified value multiplied by 1.25 ms.

param connection_interval numeric Range: 6 to 3200

7.1.4.21 Sinterval

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:SINTerval:LESignaling
```

class Sinterval

Sinterval commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_le_signaling() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:SINTerval:LESignaling
value: int = driver.configure.connection.sinterval.get_le_signaling()
```

Specifies the interval between two consecutive page scans. The interval in ms is calculated as the specified value multiplied by 0.625 ms.

return conn_scan_int: numeric Range: 4 to 16.384E+3

set_le_signaling(*conn_scan_int: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:SINTerval:LESignaling
driver.configure.connection.sinterval.set_le_signaling(conn_scan_int = 1)
```

Specifies the interval between two consecutive page scans. The interval in ms is calculated as the specified value multiplied by 0.625 ms.

param conn_scan_int numeric Range: 4 to 16.384E+3

7.1.4.22 Ainterval

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AINTerval:LESignaling
```

class Ainterval

Ainterval commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_le_signaling() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AINTerval:LESignaling
value: int = driver.configure.connection.ainterval.get_le_signaling()
```

Specifies the interval between two consecutive advertisers for an instrument in peripheral LE role.

return adv_int: numeric Range: 32 to 16.384E+3

set_le_signaling(*adv_int: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:AINTerval:LESignaling
driver.configure.connection.ainterval.set_le_signaling(adv_int = 1)
```

Specifies the interval between two consecutive advertisers for an instrument in peripheral LE role.

param adv_int numeric Range: 32 to 16.384E+3

7.1.4.23 Swindow

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:SWINdow:LESignaling
```

class Swindow

Swindow commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_le_signaling() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:SWINdow:LESignaling
value: int = driver.configure.connection.swindow.get_le_signaling()
```

Specifies the interval between two consecutive page scans. The interval in ms is calculated as the specified value multiplied by 0.625 ms.

return conn_scan_win: numeric Range: 4 to 16.384E+3

set_le_signaling(*conn_scan_win: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:SWINdow:LESignaling
driver.configure.connection.swindow.set_le_signaling(conn_scan_win = 1)
```

Specifies the interval between two consecutive page scans. The interval in ms is calculated as the specified value multiplied by 0.625 ms.

param conn_scan_win numeric Range: 4 to 16.384E+3

7.1.4.24 Pperiod

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PPERiod:MINimum
CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PPERiod
```

class Pperiod

Pperiod commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

get_minimum() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PPERiod:MINimum
value: bool = driver.configure.connection.pperiod.get_minimum()
```

Enables minimum poll period. To prevent simultaneous master/slave transmission, the minimum poll period for an x-DHn packet type (n = 1, 3, 5) is automatically set to n+1 slots.

return poll_period_min: OFF | ON

get_value() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PPERiod
value: int = driver.configure.connection.pperiod.get_value()
```

Defines how often the R&S CMW transmitts a poll packet. If the set poll period is too small for the selected packet type (for x-DH1, x-DH3, or x-DH5) , it is automatically changed to 2, 4, or 6 slots. X = 1, 2, 3.

return poll_period: numeric Range: x-DH1: 1 to 127, x-DH3: 2 to 127, x-DH5: 3 to 127 , Unit: Unit corresponds to two slots

set_minimum(*poll_period_min: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PPERiod:MINimum
driver.configure.connection.pperiod.set_minimum(poll_period_min = False)
```

Enables minimum poll period. To prevent simultaneous master/slave transmission, the minimum poll period for an x-DHn packet type (n = 1, 3, 5) is automatically set to n+1 slots.

param poll_period_min OFF | ON

set_value(*poll_period: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:CONNection:PPERiod
driver.configure.connection.pperiod.set_value(poll_period = 1)
```

Defines how often the R&S CMW transmitts a poll packet. If the set poll period is too small for the selected packet type (for x-DH1, x-DH3, or x-DH5) , it is automatically changed to 2, 4, or 6 slots. X = 1, 2, 3.

param poll_period numeric Range: x-DH1: 1 to 127, x-DH3: 2 to 127, x-DH5: 3 to 127 , Unit: Unit corresponds to two slots

7.1.5 LowEnergy

class LowEnergy
LowEnergy commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.5.1 Reset

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:LENergy:RESet:DELay
```

class Reset
Reset commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_delay() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:LENergy:RESet:DELay
value: float = driver.configure.lowEnergy.reset.get_delay()
```

Specifies a delay after EUT reset.

return delay_after_reset: numeric Range: 0 s to 0.2 s

set_delay(*delay_after_reset: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:LENergy:RESet:DELay
driver.configure.lowEnergy.reset.set_delay(delay_after_reset = 1.0)
```

Specifies a delay after EUT reset.

param delay_after_reset numeric Range: 0 s to 0.2 s

7.1.6 UsbSettings<UsbSettings>

RepCap Settings

```
# Range: Sett1 .. Sett4
rc = driver.configure.usbSettings.repcap_usbSettings_get()
driver.configure.usbSettings.repcap_usbSettings_set(repcap.UsbSettings.Sett1)
```

class UsbSettings
UsbSettings commands group definition. 2 total commands, 2 Sub-groups, 0 group commands Repeated Capability: UsbSettings, default value after init: UsbSettings.Sett1

Cloning the Group

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

Subgroups

7.1.6.1 UsbDevice

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:USBSettings<UsbSettings>:USBDevice
```

class UsbDevice

UsbDevice commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*usbSettings=<UsbSettings.Default: -1>*) → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:USBSettings<nr>:USBDevice
value: int = driver.configure.usbSettings.usbDevice.get(usbSettings = repcap.
↪UsbSettings.Default)
```

Specifies the USB port to be used for direct USB connection. The command is relevant for the direct USB connection ('HW Interface' = USB) .

param usbSettings optional repeated capability selector. Default value: Sett1 (settable in the interface 'UsbSettings')

return no: 1..2 1: HW interface for LE tests 2: HW interface for BR / EDR tests

set(*no: int*, *usbSettings=<UsbSettings.Default: -1>*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:USBSettings<nr>:USBDevice
driver.configure.usbSettings.usbDevice.set(no = 1, usbSettings = repcap.
↪UsbSettings.Default)
```

Specifies the USB port to be used for direct USB connection. The command is relevant for the direct USB connection ('HW Interface' = USB) .

param no 1..2 1: HW interface for LE tests 2: HW interface for BR / EDR tests

param usbSettings optional repeated capability selector. Default value: Sett1 (settable in the interface 'UsbSettings')

7.1.6.2 Devices

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:USBSettings:DEVices:CATalog
```

class Devices

Devices commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class CatalogStruct

Structure for reading output parameters. Fields:

- No_Devices: int: decimal Number of all USB ports, where a connected EUT has been recognized
- Item_Number: List[int]: decimal The number of a list item
- Discovered_Port: List[str]: string Number of the USB port

get_catalog() → CatalogStruct

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:USBSettings:DEVices:CATalog
value: CatalogStruct = driver.configure.usbSettings.devices.get_catalog()
```

Displays all EUTs discovered at USB interface. The command is relevant for the direct USB connection ('HW Interface' = USB) . Results are returned for each connected EUT: <NoDevices>, {1, <DiscoveredPort>}1, … , {<NoDevices>, <DiscoveredPort>}<NoDevices>

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

7.1.7 ComSettings<CommSettings>

RepCap Settings

```
# Range: Hw1 .. Hw4
rc = driver.configure.comSettings.repcap_commSettings_get()
driver.configure.comSettings.repcap_commSettings_set(repcap.CommSettings.Hw1)
```

class ComSettings

ComSettings commands group definition. 8 total commands, 8 Sub-groups, 0 group commands Repeated Capability: CommSettings, default value after init: CommSettings.Hw1

Cloning the Group

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

Subgroups

7.1.7.1 StopBits

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:COMSettings<CommSettings>:STOPbits
```

class StopBits

StopBits commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*commSettings=<CommSettings.Default: -1>*) → RsCmwBluetoothSig.enums.StopBits

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:COMSettings<nr>:STOPbits
value: enums.StopBits = driver.configure.comSettings.stopBits.get(commSettings␣
↪= repcap.CommSettings.Default)
```

Specifies the transmission parameters of serial connection.

param commSettings optional repeated capability selector. Default value: Hw1 (settable in the interface 'ComSettings')

return stop_bits: S1 | S2 Number of bits used for stop indication

set(*stop_bits: RsCmwBluetoothSig.enums.StopBits*, *commSettings=<CommSettings.Default: -1>*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:COMSettings<nr>:STOPbits
driver.configure.comSettings.stopBits.set(stop_bits = enums.StopBits.S1,␣
↪commSettings = repcap.CommSettings.Default)
```

Specifies the transmission parameters of serial connection.

param stop_bits S1 | S2 Number of bits used for stop indication

param commSettings optional repeated capability selector. Default value: Hw1 (settable in the interface 'ComSettings')

7.1.7.2 Parity

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:COMSettings<CommSettings>:PARity
```

class Parity

Parity commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*commSettings=<CommSettings.Default: -1>*) → RsCmwBluetoothSig.enums.Parity

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:COMSettings<nr>:PARity
value: enums.Parity = driver.configure.comSettings.parity.get(commSettings =␣
↪repcap.CommSettings.Default)
```

Specifies the transmission parameters of serial connection.

param commSettings optional repeated capability selector. Default value: Hw1 (settable in the interface 'ComSettings')

return parity: NONE | ODD | EVEN Number of parity bits

set(*parity: RsCmwBluetoothSig.enums.Parity*, *commSettings=<CommSettings.Default: -1>*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:COMSettings<nr>:PARity
driver.configure.comSettings.parity.set(parity = enums.Parity.EVEN,␣
↪commSettings = repcap.CommSettings.Default)
```

Specifies the transmission parameters of serial connection.

param parity NONE | ODD | EVEN Number of parity bits

param commSettings optional repeated capability selector. Default value: Hw1 (settable in the interface 'ComSettings')

7.1.7.3 Dbits

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:COMSettings<CommSettings>:DBITs
```

class Dbits

Dbits commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*commSettings=<CommSettings.Default: -1>*) → RsCmwBluetoothSig.enums.DataBits

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:COMSettings<nr>:DBITs
value: enums.DataBits = driver.configure.comSettings.dbits.get(commSettings =␣
↪repcap.CommSettings.Default)
```

No command help available

param commSettings optional repeated capability selector. Default value: Hw1 (settable in the interface 'ComSettings')

return data_bits: No help available

set(*data_bits: RsCmwBluetoothSig.enums.DataBits*, *commSettings=<CommSettings.Default: -1>*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:COMSettings<nr>:DBITs
driver.configure.comSettings.dbits.set(data_bits = enums.DataBits.D7,␣
↪commSettings = repcap.CommSettings.Default)
```

No command help available

param data_bits No help available

param commSettings optional repeated capability selector. Default value: Hw1 (settable in the interface 'ComSettings')

7.1.7.4 ComPort

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:COMSettings<CommSettings>:COMPort
```

class ComPort

ComPort commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*commSettings*=<*CommSettings.Default: -1*>) → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:COMSettings<nr>:COMPort
value: int = driver.configure.comSettings.comPort.get(commSettings = repcap.
↪CommSettings.Default)
```

Specifies the virtual COM port to be used for USB connection with USB to RS232 adapter.

param commSettings optional repeated capability selector. Default value: Hw1 (settable in the interface 'ComSettings')

return no: 1..2 1: HW interface for LE tests 2: HW interface for BR / EDR tests

set(*no: int*, *commSettings*=<*CommSettings.Default: -1*>) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:COMSettings<nr>:COMPort
driver.configure.comSettings.comPort.set(no = 1, commSettings = repcap.
↪CommSettings.Default)
```

Specifies the virtual COM port to be used for USB connection with USB to RS232 adapter.

param no 1..2 1: HW interface for LE tests 2: HW interface for BR / EDR tests

param commSettings optional repeated capability selector. Default value: Hw1 (settable in the interface 'ComSettings')

7.1.7.5 BaudRate

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:COMSettings<CommSettings>:BAUDrate
```

class BaudRate

BaudRate commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*commSettings*=<*CommSettings.Default: -1*>) → RsCmwBluetoothSig.enums.BaudRate

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:COMSettings<nr>:BAUDrate
value: enums.BaudRate = driver.configure.comSettings.baudRate.get(commSettings␣
↪= repcap.CommSettings.Default)
```

Specifies the transmission parameters of serial connection.

param commSettings optional repeated capability selector. Default value: Hw1 (settable in the interface 'ComSettings')

return baud_rate: B110 | B300 | B600 | B12K | B24K | B48K | B96K | B14K | B19K | B28K | B38K | B57K | B115k | B234k | B460k | B500k | B576k | B921k | B1M | B1M5 | B2M | B3M | B3M5 | B4M Data transmission rate in symbol: 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, 115200, 230400, 460800, 500000, 576000, 921600, 1000000, 1152000, 2000000, 3000000, 3500000, 4000000

set(*baud_rate: RsCmwBluetoothSig.enums.BaudRate*, *commSettings=<CommSettings.Default: -1>*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:COMSettings<nr>:BAUDrate
driver.configure.comSettings.baudRate.set(baud_rate = enums.BaudRate.B110,␣
↪commSettings = repcap.CommSettings.Default)
```

Specifies the transmission parameters of serial connection.

param baud_rate B110 | B300 | B600 | B12K | B24K | B48K | B96K | B14K | B19K | B28K | B38K | B57K | B115k | B234k | B460k | B500k | B576k | B921k | B1M | B1M5 | B2M | B3M | B3M5 | B4M Data transmission rate in symbol: 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, 115200, 230400, 460800, 500000, 576000, 921600, 1000000, 1152000, 2000000, 3000000, 3500000, 4000000

param commSettings optional repeated capability selector. Default value: Hw1 (settable in the interface 'ComSettings')

7.1.7.6 Protocol

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:COMSettings<CommSettings>:PROTocol
```

class Protocol

Protocol commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*commSettings=<CommSettings.Default: -1>*) → RsCmwBluetoothSig.enums.Protocol

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:COMSettings<nr>:PROTocol
value: enums.Protocol = driver.configure.comSettings.protocol.get(commSettings␣
↪= repcap.CommSettings.Default)
```

Specifies the transmission parameters of serial connection.

param commSettings optional repeated capability selector. Default value: Hw1 (settable in the interface 'ComSettings')

return protocol: XONXoff | CTSRts | NONE Transmit flow control X-ON/X-OFF, RFR/CTS, or none

set(*protocol: RsCmwBluetoothSig.enums.Protocol*, *commSettings=<CommSettings.Default: -1>*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:COMSettings<nr>:PROTocol
driver.configure.comSettings.protocol.set(protocol = enums.Protocol.CTSRts,␣
↪commSettings = repcap.CommSettings.Default)
```

Specifies the transmission parameters of serial connection.

param protocol XONXoff | CTSRts | NONE Transmit flow control X-ON/X-OFF, RFR/CTS, or none

param commSettings optional repeated capability selector. Default value: Hw1 (settable in the interface 'ComSettings')

7.1.7.7 Ereset

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:COMSettings<CommSettings>:ERESet
```

class Ereset

Ereset commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*commSettings*=<*CommSettings.Default: -1*>) → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:COMSettings<nr>:ERESet
value: bool = driver.configure.comSettings.ereset.get(commSettings = repcap.
↪CommSettings.Default)
```

Commands to reset the EUT before starting tests.

param commSettings optional repeated capability selector. Default value: Hw1 (settable in the interface 'ComSettings')

return eut_reset: ON | OFF

set(*eut_reset: bool*, *commSettings*=<*CommSettings.Default: -1*>) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:COMSettings<nr>:ERESet
driver.configure.comSettings.ereset.set(eut_reset = False, commSettings =␣
↪repcap.CommSettings.Default)
```

Commands to reset the EUT before starting tests.

param eut_reset ON | OFF

param commSettings optional repeated capability selector. Default value: Hw1 (settable in the interface 'ComSettings')

7.1.7.8 Ports

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:COMSettings:PORTs:CATalog
```

class Ports

Ports commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class CatalogStruct

Structure for reading output parameters. Fields:

- No_Devices: int: decimal Number of all COM ports, where a connected EUT has been recognized

- Item_Number: List[int]: decimal The number of a list item
- Discovered_Port: List[str]: string Number of the virtual COM port, to which the used USB port has been mapped

get_catalog() → CatalogStruct

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:COMSettings:PORTs:CATalog
value: CatalogStruct = driver.configure.comSettings.ports.get_catalog()
```

Queries the COM ports used by an EUT. The command is relevant for the USB connection with USB-to-serial converter ('HW Interface' = USB to RS232 adapter) . Results are returned for each used USB port: <NoDevices>, {1, <DiscoveredPort>}1, ... , {<NoDevices>, <DiscoveredPort>}<NoDevices>

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

7.1.8 HwInterface<HardwareIntf>

RepCap Settings

```
# Range: Intf1 .. Intf4
rc = driver.configure.hwInterface.repcap_hardwareIntf_get()
driver.configure.hwInterface.repcap_hardwareIntf_set(repcap.HardwareIntf.Intf1)
```

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:HWINterface<HardwareIntf>
```

class HwInterface

HwInterface commands group definition. 1 total commands, 0 Sub-groups, 1 group commands Repeated Capability: HardwareIntf, default value after init: HardwareIntf.Intf1

get(*hardwareIntf=<HardwareIntf.Default: -1>*) → RsCmwBluetoothSig.enums.HwInterface

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:HWINterface<nr>
value: enums.HwInterface = driver.configure.hwInterface.get(hardwareIntf =␣
↪repcap.HardwareIntf.Default)
```

Defines interface used for tests.

param hardwareIntf optional repeated capability selector. Default value: Intf1 (settable in the interface 'HwInterface')

return hw_interface: NONE | RS232 | USB RS232: USB connection with USB to RS232 adapter NONE: no control via USB to be used USB: direct USB connection

set(*hw_interface: RsCmwBluetoothSig.enums.HwInterface*, *hardwareIntf=<HardwareIntf.Default: -1>*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:HWINterface<nr>
driver.configure.hwInterface.set(hw_interface = enums.HwInterface.NONE,␣
↪hardwareIntf = repcap.HardwareIntf.Default)
```

Defines interface used for tests.

param hw_interface NONE | RS232 | USB RS232: USB connection with USB to RS232 adapter NONE: no control via USB to be used USB: direct USB connection

param hardwareIntf optional repeated capability selector. Default value: Intf1 (settable in the interface 'HwInterface')

Cloning the Group

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

7.1.9 Debug

class Debug

Debug commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.9.1 Rx

class Rx

Rx commands group definition. 3 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.9.1.1 Correlation

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:DEBug:RX:CORRelation:THReshold
CONFigure:BLUetooth:SIGNaling<Instance>:DEBug:RX:CORRelation:TIMeout
```

class Correlation

Correlation commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

get_threshold() → str

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:DEBug:RX:CORRelation:THReshold
value: str = driver.configure.debug.rx.correlation.get_threshold()
```

No command help available

return threshold: No help available

get_timeout() → str

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:DEBug:RX:CORRelation:TIMeout
value: str = driver.configure.debug.rx.correlation.get_timeout()
```

No command help available

return timeout: No help available

set_threshold(*threshold: str*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:DEBug:RX:CORRelation:THReshold
driver.configure.debug.rx.correlation.set_threshold(threshold = r1)
```

No command help available

param threshold No help available

set_timeout(*timeout: str*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:DEBug:RX:CORRelation:TIMeout
driver.configure.debug.rx.correlation.set_timeout(timeout = r1)
```

No command help available

param timeout No help available

7.1.9.1.2 Trigger

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:DEBug:RX:TRIGger:PLEVel
```

class Trigger

Trigger commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_plevel() → str

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:DEBug:RX:TRIGger:PLEVel
value: str = driver.configure.debug.rx.trigger.get_plevel()
```

No command help available

return trigger: No help available

set_plevel(*trigger: str*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:DEBug:RX:TRIGger:PLEVel
driver.configure.debug.rx.trigger.set_plevel(trigger = r1)
```

No command help available

param trigger No help available

7.1.10 Tconnection

class Tconnection

Tconnection commands group definition. 11 total commands, 6 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.10.1 Interval

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:TCONnection:INTerval:LESignaling
```

class Interval

Interval commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_le_signaling() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:TCONnection:INTerval:LESignaling
value: int = driver.configure.tconnection.interval.get_le_signaling()
```

Sets the time interval between the two consecutive connection events for LE test mode. The interval in ms is calculated as the specified value multiplied by 1.25 ms.

return connection_interval: numeric Range: 6 to 3200

set_le_signaling(*connection_interval: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:TCONnection:INTerval:LESignaling
driver.configure.tconnection.interval.set_le_signaling(connection_interval = 1)
```

Sets the time interval between the two consecutive connection events for LE test mode. The interval in ms is calculated as the specified value multiplied by 1.25 ms.

param connection_interval numeric Range: 6 to 3200

7.1.10.2 SpinEnable

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:TCONnection:SPINenable:LENergy
```

class SpinEnable

SpinEnable commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_low_energy() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:TCONnection:SPINenable:LENergy
value: bool = driver.configure.tconnection.spinEnable.get_low_energy()
```

Enables or disables LE test mode on the DUT using a PIN. The PIN to sent is specified via: method RsCmwBluetoothSig. Configure.Tconnection.SpinEnable.lowEnergy

return send_enable_pin: OFF | ON

set_low_energy(*send_enable_pin: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:TCONnection:SPINenable:LENergy
driver.configure.tconnection.spinEnable.set_low_energy(send_enable_pin = False)
```

Enables or disables LE test mode on the DUT using a PIN. The PIN to sent is specified via: method RsCmwBluetoothSig. Configure.Tconnection.SpinEnable.lowEnergy

param send_enable_pin OFF | ON

7.1.10.3 PinCode

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:TCONnection:PINCode:LENergy
```

class PinCode

PinCode commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_low_energy() → List[int]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:TCONnection:PINCode:LENergy
value: List[int] = driver.configure.tconnection.pinCode.get_low_energy()
```

Specifies the PIN for LE test mode. The value must match with the configuration on the DUT.

return pin: integer Comma-separated sequence of eight integers Range: 0 to 255

set_low_energy(*pin: List[int]*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:TCONnection:PINCode:LENergy
driver.configure.tconnection.pinCode.set_low_energy(pin = [1, 2, 3])
```

Specifies the PIN for LE test mode. The value must match with the configuration on the DUT.

param pin integer Comma-separated sequence of eight integers Range: 0 to 255

7.1.10.4 Packets

class Packets
Packets commands group definition. 6 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.10.4.1 Pattern

class Pattern
Pattern commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.10.4.1.1 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:TCONnection:PACKets:PATTern:LENergy:LE1M
CONFigure:BLUetooth:SIGNaling<Instance>:TCONnection:PACKets:PATTern:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>:TCONnection:PACKets:PATTern:LENergy:LRANge
```

class LowEnergy
LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → RsCmwBluetoothSig.enums.LeRangePaternType

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:TCONnection:PACKets:PATTern:LENergy:LE1M
value: enums.LeRangePaternType = driver.configure.tconnection.packets.pattern.
↪lowEnergy.get_le_1_m()
```

Select the bit pattern to be used for tests. INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Commands for test mode classic: For BR (...:BRATe...) , EDR (...:EDRate...)

- Commands for LE direct test mode: For LE 1M PHY (...:LE1M...) , LE 2M PHY (...:LE2M...)
- Commands for LE test mode: ..:TCONnection:..

return pattern_type: ALL0 | ALL1 | P11 | P44 | PRBS9 | ALT ALL0: 00000000 ALL1: 11111111 P11: 10101010 P44: 11110000 PRBS9: pseudo-random bit sequences of a length of 9 bits (transmission of identical packet series) ALT: the periodical alternation of the pattern P11 and P44

get_le_2_m() → RsCmwBluetoothSig.enums.LeRangePaternType

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:TCONnection:PACKets:PATTern:LENergy:LE2M
value: enums.LeRangePaternType = driver.configure.tconnection.packets.pattern.
↪lowEnergy.get_le_2_m()
```

Select the bit pattern to be used for tests. INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Commands for test mode classic: For BR (...:BRATe...) , EDR (...:EDRate...)
- Commands for LE direct test mode: For LE 1M PHY (...:LE1M...) , LE 2M PHY (...:LE2M...)
- Commands for LE test mode: ..:TCONnection:..

return pattern_type: ALL0 | ALL1 | P11 | P44 | PRBS9 | ALT ALL0: 00000000 ALL1: 11111111 P11: 10101010 P44: 11110000 PRBS9: pseudo-random bit sequences of a length of 9 bits (transmission of identical packet series) ALT: the periodical alternation of the pattern P11 and P44

get_lrange() → RsCmwBluetoothSig.enums.LeRangePaternType

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:TCONnection:PACKets:PATTern:LENergy:LRANge
value: enums.LeRangePaternType = driver.configure.tconnection.packets.pattern.
↪lowEnergy.get_lrange()
```

Select the bit pattern to be used for tests. INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Commands for test mode classic: For BR (...:BRATe...) , EDR (...:EDRate...)
- Commands for LE direct test mode: For LE 1M PHY (...:LE1M...) , LE 2M PHY (...:LE2M...)
- Commands for LE test mode: ..:TCONnection:..

return pattern_type: ALL0 | ALL1 | P11 | P44 | PRBS9 | ALT ALL0: 00000000 ALL1: 11111111 P11: 10101010 P44: 11110000 PRBS9: pseudo-random bit sequences of a length of 9 bits (transmission of identical packet series) ALT: the periodical alternation of the pattern P11 and P44

set_le_1_m(*pattern_type: RsCmwBluetoothSig.enums.LeRangePaternType*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:TCONnection:PACKets:PATTern:LENergy:LE1M
driver.configure.tconnection.packets.pattern.lowEnergy.set_le_1_m(pattern_type␣
↪= enums.LeRangePaternType.ALL0)
```

Select the bit pattern to be used for tests. INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Commands for test mode classic: For BR (...:BRATe...) , EDR (...:EDRate...)
- Commands for LE direct test mode: For LE 1M PHY (...:LE1M...) , LE 2M PHY (...:LE2M...)
- Commands for LE test mode: ..:TCONnection:..

param pattern_type ALL0 | ALL1 | P11 | P44 | PRBS9 | ALT ALL0: 00000000 ALL1: 11111111 P11: 10101010 P44: 11110000 PRBS9: pseudo-random bit sequences of a length of 9 bits (transmission of identical packet series) ALT: the periodical alternation of the pattern P11 and P44

set_le_2_m(*pattern_type: RsCmwBluetoothSig.enums.LeRangePaternType*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:TCONnection:PACKets:PATTern:LENergy:LE2M
driver.configure.tconnection.packets.pattern.lowEnergy.set_le_2_m(pattern_type␣
↪= enums.LeRangePaternType.ALL0)
```

Select the bit pattern to be used for tests. INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Commands for test mode classic: For BR (...:BRATe...) , EDR (...:EDRate...)
- Commands for LE direct test mode: For LE 1M PHY (...:LE1M...) , LE 2M PHY (...:LE2M...)
- Commands for LE test mode: ..:TCONnection:..

param pattern_type ALL0 | ALL1 | P11 | P44 | PRBS9 | ALT ALL0: 00000000 ALL1: 11111111 P11: 10101010 P44: 11110000 PRBS9: pseudo-random bit sequences of a length of 9 bits (transmission of identical packet series) ALT: the periodical alternation of the pattern P11 and P44

set_lrange(*pattern_type: RsCmwBluetoothSig.enums.LeRangePaternType*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:TCONnection:PACKets:PATTern:LENergy:LRANge
driver.configure.tconnection.packets.pattern.lowEnergy.set_lrange(pattern_type␣
↪= enums.LeRangePaternType.ALL0)
```

Select the bit pattern to be used for tests. INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Commands for test mode classic: For BR (...:BRATe...) , EDR (...:EDRate...)
- Commands for LE direct test mode: For LE 1M PHY (...:LE1M...) , LE 2M PHY (...:LE2M...)
- Commands for LE test mode: ..:TCONnection:..

param pattern_type ALL0 | ALL1 | P11 | P44 | PRBS9 | ALT ALL0: 00000000 ALL1: 11111111 P11: 10101010 P44: 11110000 PRBS9: pseudo-random bit sequences of a length of 9 bits (transmission of identical packet series) ALT: the periodical alternation of the pattern P11 and P44

7.1.10.4.2 PacketLength

class PacketLength

PacketLength commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.10.4.2.1 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:TCONnection:PACKets:PLENgth:LENergy:LE1M
CONFigure:BLUetooth:SIGNaling<Instance>:TCONnection:PACKets:PLENgth:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>:TCONnection:PACKets:PLENgth:LENergy:LRANge
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:TCONnection:PACKets:PLENgth:LENergy:LE1M
value: int = driver.configure.tconnection.packets.packetLength.lowEnergy.get_le_
↪1_m()
```

Specifies the payload length. INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Commands for LE direct test mode: For LE 1M PHY (...:LE1M...) , LE 2M PHY (...:LE2M...) , and LE coded PHY (...:LRANge...) .

INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Commands for LE test mode: ..:TCONnection:..

return payload_length: numeric Range: 0 byte(s) to 255 byte(s) , Unit: byte

get_le_2_m() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:TCONnection:PACKets:PLENgth:LENergy:LE2M
value: int = driver.configure.tconnection.packets.packetLength.lowEnergy.get_le_
↪2_m()
```

Specifies the payload length. INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Commands for LE direct test mode: For LE 1M PHY (...:LE1M...) , LE 2M PHY (...:LE2M...) , and LE coded PHY (...:LRANge...) .

INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Commands for LE test mode: ..:TCONnection:..

return payload_length: numeric Range: 0 byte(s) to 255 byte(s) , Unit: byte

get_lrange() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:TCONnection:PACKets:PLENgth:LENergy:LRANge
value: int = driver.configure.tconnection.packets.packetLength.lowEnergy.get_
↪lrange()
```

Specifies the payload length. INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Commands for LE direct test mode: For LE 1M PHY (...:LE1M...) , LE 2M PHY (...:LE2M...) , and LE coded PHY (...:LRANge...) .

INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Commands for LE test mode: ..:TCONnection:..

return payload_length: numeric Range: 0 byte(s) to 255 byte(s) , Unit: byte

set_le_1_m(*payload_length: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:TCONnection:PACKets:PLENgth:LENergy:LE1M
driver.configure.tconnection.packets.packetLength.lowEnergy.set_le_1_m(payload_
↪length = 1)
```

Specifies the payload length. INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Commands for LE direct test mode: For LE 1M PHY (...:LE1M...) , LE 2M PHY (...:LE2M...) , and LE coded PHY (...:LRANge...) .

INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Commands for LE test mode: ..:TCONnection:..

param payload_length numeric Range: 0 byte(s) to 255 byte(s) , Unit: byte

set_le_2_m(*payload_length: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:TCONnection:PACKets:PLENgth:LENergy:LE2M
driver.configure.tconnection.packets.packetLength.lowEnergy.set_le_2_m(payload_
↪length = 1)
```

Specifies the payload length. INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Commands for LE direct test mode: For LE 1M PHY (...:LE1M...) , LE 2M PHY (...:LE2M...) , and LE coded PHY (...:LRANge...) .

INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Commands for LE test mode: ..:TCONnection:..

param payload_length numeric Range: 0 byte(s) to 255 byte(s) , Unit: byte

set_lrange(*payload_length: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:TCONnection:PACKets:PLENgth:LENergy:LRANge
driver.configure.tconnection.packets.packetLength.lowEnergy.set_lrange(payload_
↪length = 1)
```

Specifies the payload length. INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Commands for LE direct test mode: For LE 1M PHY (...:LE1M...) , LE 2M PHY (...:LE2M...) , and LE coded PHY (...:LRANge...) .

INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Commands for LE test mode: ..:TCONnection:..

param payload_length numeric Range: 0 byte(s) to 255 byte(s) , Unit: byte

7.1.10.5 Phy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:TCONnection:PHY:LENergy
```

class Phy

Phy commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_low_energy() → RsCmwBluetoothSig.enums.LePhysicalType

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:TCONnection:PHY:LENergy
value: enums.LePhysicalType = driver.configure.tconnection.phy.get_low_energy()
```

Selects the physical layer used for LE connections. INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Command for LE direct test mode: ..:PHY:LENergy

INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Command for LE connection tests (normal mode) : ..:PHY:NMODe:LENergy

INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Command for LE test mode: ..:TCONnection:PHY:LENergy

return phy: LE1M | LE2M | LELR LE1M: LE 1 Msymbol/s uncoded PHY LE2M: LE 2 Msymbol/s uncoded PHY LELR: LE 1 Msymbol/s long range (LE coded PHY)

set_low_energy(*phy: RsCmwBluetoothSig.enums.LePhysicalType*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:TCONnection:PHY:LENergy
driver.configure.tconnection.phy.set_low_energy(phy = enums.LePhysicalType.LE1M)
```

Selects the physical layer used for LE connections. INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Command for LE direct test mode: ..:PHY:LENergy

INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Command for LE connection tests (normal mode) : ..:PHY:NMODe:LENergy

INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Command for LE test mode: ..:TCONnection:PHY:LENergy

param phy LE1M | LE2M | LELR LE1M: LE 1 Msymbol/s uncoded PHY LE2M: LE 2 Msymbol/s uncoded PHY LELR: LE 1 Msymbol/s long range (LE coded PHY)

7.1.10.6 Fec

class Fec

Fec commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.10.6.1 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:TCONnection:FEC:LENergy:LRANge
```

class LowEnergy

LowEnergy commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_lrange() → RsCmwBluetoothSig.enums.CodingScheme

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:TCONnection:FEC:LENergy:LRANge
value: enums.CodingScheme = driver.configure.tconnection.fec.lowEnergy.get_
↪lrange()
```

Defines the coding S for LE coded PHY according to the core specification version 5.0 for Bluetooth wireless technolo

INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Command for LE direct test mode: ..:FEC:LENergy..

INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Command for LE connection tests (normal mode) : ..:FEC:NMODe:LENergy..

INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Command for LE test mode: ..:TCONnection:FEC:LENergy..

return coding_scheme: S8 | S2 Coding S = 8 or S = 2

set_lrange(*coding_scheme: RsCmwBluetoothSig.enums.CodingScheme*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:TCONnection:FEC:LENergy:LRANge
driver.configure.tconnection.fec.lowEnergy.set_lrange(coding_scheme = enums.
↪CodingScheme.S2)
```

Defines the coding S for LE coded PHY according to the core specification version 5.0 for Bluetooth wireless technolo

INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Command for LE direct test mode: ..:FEC:LENergy..

INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Command for LE connection tests (normal mode) : ..:FEC:NMODe:LENergy..

INTRO_CMD_HELP: Defines the Bluetooth burst type. The command is relevant for:

- Command for LE test mode: ..:TCONnection:FEC:LENergy..

param coding_scheme S8 | S2 Coding S = 8 or S = 2

7.1.11 RfSettings

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:ARPower
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:ARANging
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:ENPower
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:LEVel
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:UMARgin
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:PCONtrol
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:HOPPing
```

class RfSettings

RfSettings commands group definition. 156 total commands, 10 Sub-groups, 7 group commands

get_ar_power() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:ARPower
value: float = driver.configure.rfSettings.get_ar_power()
```

No command help available

return inp_level: No help available

get_aranging() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<instance>:RFSettings:ARANging
value: bool = driver.configure.rfSettings.get_aranging()
```

Adjusts the input level expected at the R&S CMW antenna automatically, according to the predefined values and measured signal amplitude.

return auto_ranging: OFF | ON

get_envelope_power() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:ENPower
value: float = driver.configure.rfSettings.get_envelope_power()
```

Sets the expected nominal power of the EUT signal

return exp_nominal_power: 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_hopping() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:HOPPing
value: bool = driver.configure.rfSettings.get_hopping()
```

Enables/disables frequency hopping.

return hopping: OFF | ON

get_level() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:LEVel
value: float = driver.configure.rfSettings.get_level()
```

Defines the absolute TX level of the R&S CMW (master) signal. The allowed value range can be calculated as follows: Range (Level) = Range (Output Power) - External Attenuation Range (Output Power) = -130 dBm to 0 dBm (RFx COM) or -120 dBm to 8 dBm (RFx OUT) ; please also notice the ranges quoted in the data sheet.

return level: numeric Range: see above , Unit: dBm

get_power_control() → RsCmwBluetoothSig.enums.PowerControl

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:PCONtrol
value: enums.PowerControl = driver.configure.rfSettings.get_power_control()
```

No command help available

return pcontrol: No help available

get_umargin() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<instance>:RFSettings:UMARgin
value: float = driver.configure.rfSettings.get_umargin()
```

Sets the margin that the R&S CMW adds to the expected nominal power to determine the reference level. The reference level minus the external input attenuation must be within the power range of the selected input connector; refer to the data sheet.

INTRO_CMD_HELP: Refer also to the following commands:

- method RsCmwBluetoothSig.Configure.RfSettings.envelopePower
- method RsCmwBluetoothSig.Configure.RfSettings.Eattenuation.inputPy

return margin: numeric Range: 0 dB to (55 dB + external attenuation - expected nominal power)

set_aranging(*auto_ranging: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<instance>:RFSettings:ARANging
driver.configure.rfSettings.set_aranging(auto_ranging = False)
```

Adjusts the input level expected at the R&S CMW antenna automatically, according to the predefined values and measured signal amplitude.

param auto_ranging OFF | ON

set_envelope_power(*exp_nominal_power: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:ENPower
driver.configure.rfSettings.set_envelope_power(exp_nominal_power = 1.0)
```

Sets the expected nominal power of the EUT signal

param exp_nominal_power 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_hopping(*hopping: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:HOPPing
driver.configure.rfSettings.set_hopping(hopping = False)
```

Enables/disables frequency hopping.

param hopping OFF | ON

set_level(*level: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:LEVel
driver.configure.rfSettings.set_level(level = 1.0)
```

Defines the absolute TX level of the R&S CMW (master) signal. The allowed value range can be calculated as follows: Range (Level) = Range (Output Power) - External Attenuation Range (Output Power) = -130 dBm to 0 dBm (RFx COM) or -120 dBm to 8 dBm (RFx OUT) ; please also notice the ranges quoted in the data sheet.

param level numeric Range: see above , Unit: dBm

set_power_control(*pcontrol: RsCmwBluetoothSig.enums.PowerControl*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:PCONtrol
driver.configure.rfSettings.set_power_control(pcontrol = enums.PowerControl.
↪DOWN)
```

No command help available

param pcontrol No help available

set_umargin(*margin: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<instance>:RFSettings:UMARgin
driver.configure.rfSettings.set_umargin(margin = 1.0)
```

Sets the margin that the R&S CMW adds to the expected nominal power to determine the reference level. The reference level minus the external input attenuation must be within the power range of the selected input connector; refer to the data sheet.

INTRO_CMD_HELP: Refer also to the following commands:

- method RsCmwBluetoothSig.Configure.RfSettings.envelopePower
- method RsCmwBluetoothSig.Configure.RfSettings.Eattenuation.inputPy

param margin numeric Range: 0 dB to (55 dB + external attenuation - expected nominal power)

Cloning the Group

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

Subgroups

7.1.11.1 Dtx

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX
```

class Dtx

Dtx commands group definition. 129 total commands, 5 Sub-groups, 1 group commands

get_value() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX
value: bool = driver.configure.rfSettings.dtx.get_value()
```

Enables/disables the dirty transmitter.

return dtx_state: OFF | ON

set_value(*dtx_state: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX
driver.configure.rfSettings.dtx.set_value(dtx_state = False)
```

Enables/disables the dirty transmitter.

param dtx_state OFF | ON

Cloning the Group

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

Subgroups

7.1.11.1.1 Stab

class Stab

Stab commands group definition. 51 total commands, 4 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.11.1.1.1 Mindex

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:MINDex:BRATe
```

class Mindex

Mindex commands group definition. 18 total commands, 5 Sub-groups, 1 group commands

get_brate() → List[float]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:MINDex:BRATe
value: List[float or bool] = driver.configure.rfSettings.dtx.stab.mindex.get_
↪brate()
```

Return the modulation index under the periodic change according to the test specification for Bluetooth wireless technology (10 values) .

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for BR (..:BRATe..) , LE 1M PHY - uncoded (..:LE1M..) , LE 2M PHY - uncoded (..:LE2M..) , and LE coded PHY (.. :LRANge..) are available. For dirty transmitter parameters according to the test specification for Bluetooth wireless technology, see also 'Dirty Tx Mode'.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return mod_index: float | ON | OFF Range: 0.2 to 0.55

Cloning the Group

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

Subgroups

7.1.11.1.1.2 Nmode

class Nmode
Nmode commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.11.1.1.3 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:MINDex:NMODe:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:MINDex:NMODe:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:MINDex:NMODe:LENergy:LE1M
```

class LowEnergy
LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → List[float]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:MINDex:NMODe:LENergy:LE1M
value: List[float or bool] = driver.configure.rfSettings.dtx.stab.mindex.nmode.
↪lowEnergy.get_le_1_m()
```

Return the modulation index under the periodic change according to the test specification for Bluetooth wireless technology (10 values) .

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for BR (..:BRATe..) , LE 1M PHY - uncoded (..:LE1M..) , LE 2M PHY - uncoded (..:LE2M..) , and LE coded PHY (.. :LRANge..) are available. For dirty transmitter parameters according to the test specification for Bluetooth wireless technology, see also 'Dirty Tx Mode'.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return mod_index: float | ON | OFF Range: 0.2 to 0.55

get_le_2_m() → List[float]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:MINDex:NMODe:LENergy:LE2M
value: List[float or bool] = driver.configure.rfSettings.dtx.stab.mindex.nmode.
↪lowEnergy.get_le_2_m()
```

Return the modulation index under the periodic change according to the test specification for Bluetooth wireless technology (10 values) .

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for BR (..:BRATe..) , LE 1M PHY - uncoded (..:LE1M..) , LE 2M PHY - uncoded (..:LE2M..) , and LE coded PHY (.. :LRANge..) are available. For dirty transmitter parameters according to the test specification for Bluetooth wireless technology, see also 'Dirty Tx Mode'.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return mod_index: float | ON | OFF Range: 0.2 to 0.55

get_lrange() → List[float]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:MINDex:NMODe:LENergy:LRANge
value: List[float or bool] = driver.configure.rfSettings.dtx.stab.mindex.nmode.
↪lowEnergy.get_lrange()
```

Return the modulation index under the periodic change according to the test specification for Bluetooth wireless technology (10 values) .

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for BR (..:BRATe..) , LE 1M PHY - uncoded (..:LE1M..) , LE 2M PHY - uncoded (..:LE2M..) , and LE coded PHY (.. :LRANge..) are available. For dirty transmitter parameters according to the test specification for Bluetooth wireless technology, see also 'Dirty Tx Mode'.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return mod_index: float | ON | OFF Range: 0.2 to 0.55

7.1.11.1.1.4 Standard

class Standard

Standard commands group definition. 6 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.11.1.1.5 Tmode

class Tmode

Tmode commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.rfSettings.dtx.stab.mindex.standard.tmode.clone()
```

Subgroups

7.1.11.1.1.6 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:MINDex:STANdard:TMODe:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:MINDex:STANdard:TMODe:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:MINDex:STANdard:TMODe:LENergy:LE1M
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → List[float]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:MINDex:STANdard:TMODe:LENergy:LE1M
value: List[float or bool] = driver.configure.rfSettings.dtx.stab.mindex.
↪standard.tmode.lowEnergy.get_le_1_m()
```

Return the modulation index under the periodic change according to the test specification for Bluetooth wireless technology (10 values) .

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for BR (..:BRATe..) , LE 1M PHY - uncoded (..:LE1M..) , LE 2M PHY - uncoded (..:LE2M..) , and LE coded PHY (.. :LRANge..) are available. For dirty transmitter parameters according to the test specification for Bluetooth wireless technology, see also 'Dirty Tx Mode'.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return mod_index: float | ON | OFF Range: 0.2 to 0.55

get_le_2_m() → List[float]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:MINDex:STANdard:TMODe:LENergy:LE2M
value: List[float or bool] = driver.configure.rfSettings.dtx.stab.mindex.
↪standard.tmode.lowEnergy.get_le_2_m()
```

Return the modulation index under the periodic change according to the test specification for Bluetooth wireless technology (10 values) .

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for BR (..:BRATe..) , LE 1M PHY - uncoded (..:LE1M..) , LE 2M PHY - uncoded (..:LE2M..) , and LE coded PHY (.. :LRANge..) are available. For dirty transmitter parameters according to the test specification for Bluetooth wireless technology, see also 'Dirty Tx Mode'.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return mod_index: float | ON | OFF Range: 0.2 to 0.55

get_lrange() → List[float]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:MINDex:STANdard:TMODe:LENergy:LRANge
value: List[float or bool] = driver.configure.rfSettings.dtx.stab.mindex.
↪standard.tmode.lowEnergy.get_lrange()
```

Return the modulation index under the periodic change according to the test specification for Bluetooth wireless technology (10 values) .

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for BR (..:BRATe..) , LE 1M PHY - uncoded (..:LE1M..) , LE 2M PHY - uncoded (..:LE2M..) , and LE coded PHY (.. :LRANge..) are available. For dirty transmitter parameters according to the test specification for Bluetooth wireless technology, see also 'Dirty Tx Mode'.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return mod_index: float | ON | OFF Range: 0.2 to 0.55

7.1.11.1.1.7 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:MINDex:STANdard:LENergy:LE1M
CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:MINDex:STANdard:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:MINDex:STANdard:LENergy:LE2M
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → List[float]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:MINDex:STANdard:LENergy[:LE1M]
value: List[float or bool] = driver.configure.rfSettings.dtx.stab.mindex.
↪standard.lowEnergy.get_le_1_m()
```

Return the modulation index under the periodic change according to the test specification for Bluetooth wireless technology (10 values) .

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for BR (..:BRATe..) , LE 1M PHY - uncoded (..:LE1M..) , LE 2M PHY - uncoded (..:LE2M..) , and LE coded PHY (.. :LRANge..) are available. For dirty transmitter parameters according to the test specification for Bluetooth wireless technology, see also 'Dirty Tx Mode'.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return mod_index: float | ON | OFF Range: 0.2 to 0.55

get_le_2_m() → List[float]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:MINDex:STANdard:LENergy:LE2M
value: List[float or bool] = driver.configure.rfSettings.dtx.stab.mindex.
↪standard.lowEnergy.get_le_2_m()
```

Return the modulation index under the periodic change according to the test specification for Bluetooth wireless technology (10 values) .

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for BR (..:BRATe..) , LE 1M PHY - uncoded (..:LE1M..) , LE 2M PHY - uncoded (..:LE2M..) , and LE coded PHY (.. :LRANge..) are available. For dirty transmitter parameters according to the test specification for Bluetooth wireless technology, see also 'Dirty Tx Mode'.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return mod_index: float | ON | OFF Range: 0.2 to 0.55

get_lrange() → List[float]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:MINDex:STANdard:LENergy:LRANge
value: List[float or bool] = driver.configure.rfSettings.dtx.stab.mindex.
↪standard.lowEnergy.get_lrange()
```

Return the modulation index under the periodic change according to the test specification for Bluetooth wireless technology (10 values) .

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for BR (..:BRATe..) , LE 1M PHY - uncoded (..:LE1M..) , LE 2M PHY - uncoded (..:LE2M..) , and LE coded PHY (.. :LRANge..) are available. For dirty transmitter parameters according to the test specification for Bluetooth wireless technology, see also 'Dirty Tx Mode'.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return mod_index: float | ON | OFF Range: 0.2 to 0.55

7.1.11.1.1.8 Stable

class Stable
Stable commands group definition. 6 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.11.1.1.9 Tmode

class Tmode
Tmode commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.rfSettings.dtx.stab.mindex.stable.tmode.clone()
```

Subgroups

7.1.11.1.1.10 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:MINDex:STABle:TMODe:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:MINDex:STABle:TMODe:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:MINDex:STABle:TMODe:LENergy:LE1M
```

class LowEnergy
LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → List[float]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:MINDex:STABle:TMODe:LENergy:LE1M
value: List[float or bool] = driver.configure.rfSettings.dtx.stab.mindex.stable.
↪tmode.lowEnergy.get_le_1_m()
```

Return the modulation index h under the periodic change (10 values) for stable range h = 0.495 to 0.505.
INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode:

Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available. For dirty transmitter parameters according to the test specification for Bluetooth wireless technology, see also 'Dirty Tx Mode'.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return mod_index: float | ON | OFF Range: 0.495 to 0.505

get_le_2_m() → List[float]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:MINDex:STABle:TMODe:LENergy:LE2M
value: List[float or bool] = driver.configure.rfSettings.dtx.stab.mindex.stable.
↪tmode.lowEnergy.get_le_2_m()
```

Return the modulation index h under the periodic change (10 values) for stable range h = 0.495 to 0.505.

INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode:

Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available. For dirty transmitter parameters according to the test specification for Bluetooth wireless technology, see also 'Dirty Tx Mode'.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return mod_index: float | ON | OFF Range: 0.495 to 0.505

get_lrange() → List[float]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:MINDex:STABle:TMODe:LENergy:LRANge
value: List[float or bool] = driver.configure.rfSettings.dtx.stab.mindex.stable.
↪tmode.lowEnergy.get_lrange()
```

Return the modulation index h under the periodic change (10 values) for stable range h = 0.495 to 0.505.

INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode:

Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available. For dirty transmitter parameters according to the test specification for Bluetooth wireless technology, see also 'Dirty Tx Mode'.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return mod_index: float | ON | OFF Range: 0.495 to 0.505

7.1.11.1.1.11 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:MINDex:STABle:LENergy:LE1M
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:MINDex:STABle:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:MINDex:STABle:LENergy:LE2M
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → List[float]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:MINDex:STABle:LENergy[:LE1M]
value: List[float or bool] = driver.configure.rfSettings.dtx.stab.mindex.stable.
↪lowEnergy.get_le_1_m()
```

Return the modulation index h under the periodic change (10 values) for stable range h = 0.495 to 0.505.

INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode:

Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available. For dirty transmitter parameters according to the test specification for Bluetooth wireless technology, see also 'Dirty Tx Mode'.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return mod_index: float | ON | OFF Range: 0.495 to 0.505

get_le_2_m() → List[float]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:MINDex:STABle:LENergy:LE2M
value: List[float or bool] = driver.configure.rfSettings.dtx.stab.mindex.stable.
↪lowEnergy.get_le_2_m()
```

Return the modulation index h under the periodic change (10 values) for stable range h = 0.495 to 0.505.

INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode:

Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available. For dirty transmitter parameters according to the test specification for Bluetooth wireless technology, see also 'Dirty Tx Mode'.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return mod_index: float | ON | OFF Range: 0.495 to 0.505

get_lrange() → List[float]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:MINDex:STABle:LENergy:LRANge
value: List[float or bool] = driver.configure.rfSettings.dtx.stab.mindex.stable.
↪lowEnergy.get_lrange()
```

Return the modulation index h under the periodic change (10 values) for stable range h = 0.495 to 0.505.

INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode:

Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available. For dirty transmitter parameters according to the test specification for Bluetooth wireless technology, see also 'Dirty Tx Mode'.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return mod_index: float | ON | OFF Range: 0.495 to 0.505

7.1.11.1.1.12 Tmode

class Tmode

Tmode commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.11.1.1.13 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:MINDex:TMODe:LENergy:LE1M
```

class LowEnergy

LowEnergy commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_le_1_m() → List[float]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:MINDex:TMODe:LENergy:LE1M
value: List[float or bool] = driver.configure.rfSettings.dtx.stab.mindex.tmode.
↪lowEnergy.get_le_1_m()
```

No command help available

return mod_index: No help available

7.1.11.1.1.14 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:MINDex:LENergy:LE1M
```

class LowEnergy

LowEnergy commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_le_1_m() → List[float]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:MINDex:LENergy[:LE1M]
value: List[float or bool] = driver.configure.rfSettings.dtx.stab.mindex.
↪lowEnergy.get_le_1_m()
```

No command help available

return mod_index: No help available

7.1.11.1.1.15 StError

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:STERror:EDRate
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:STERror:BRATe
```

class StError

StError commands group definition. 11 total commands, 3 Sub-groups, 2 group commands

get_brate() → List[int]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:STERror:BRATe
value: List[int] = driver.configure.rfSettings.dtx.stab.stError.get_brate()
```

Return the symbol timing error under the periodic change according to the test specification for Bluetooth wireless technology (10 values for BR and LE, 3 values for EDR) .

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available. For dirty transmitter parameters according to the test specification for Bluetooth wireless technology, see also 'Dirty Tx Mode'.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return symbol_time_err: decimal Range: - 50 ppm to 50 ppm, for BR/EDR: - 20 ppm to 20 ppm , Unit: ppm

get_edrate() → List[int]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:STERror:EDRate
value: List[int] = driver.configure.rfSettings.dtx.stab.stError.get_edrate()
```

Return the symbol timing error under the periodic change according to the test specification for Bluetooth wireless technology (10 values for BR and LE, 3 values for EDR) .

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available. For dirty transmitter parameters according to the test specification for Bluetooth wireless technology, see also 'Dirty Tx Mode'.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return symbol_time_err: decimal Range: - 50 ppm to 50 ppm, for BR/EDR: - 20 ppm to 20 ppm , Unit: ppm

Cloning the Group

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

Subgroups

7.1.11.1.1.16 Nmode

class Nmode

Nmode commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.11.1.1.17 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:STERror:NMODe:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:STERror:NMODe:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:STERror:NMODe:LENergy:LE1M
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → List[int]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:STERror:NMODe:LENergy:LE1M
value: List[int] = driver.configure.rfSettings.dtx.stab.stError.nmode.lowEnergy.
↪get_le_1_m()
```

Return the symbol timing error under the periodic change according to the test specification for Bluetooth wireless technology (10 values for BR and LE, 3 values for EDR) .

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available. For dirty transmitter parameters according to the test specification for Bluetooth wireless technology, see also 'Dirty Tx Mode'.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return symbol_time_err: decimal Range: - 50 ppm to 50 ppm, for BR/EDR: - 20 ppm to 20 ppm , Unit: ppm

get_le_2_m() → List[int]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:STERror:NMODe:LENergy:LE2M
value: List[int] = driver.configure.rfSettings.dtx.stab.stError.nmode.lowEnergy.
↪get_le_2_m()
```

Return the symbol timing error under the periodic change according to the test specification for Bluetooth wireless technology (10 values for BR and LE, 3 values for EDR) .

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available. For dirty transmitter parameters according to the test specification for Bluetooth wireless technology, see also 'Dirty Tx Mode'.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return symbol_time_err: decimal Range: - 50 ppm to 50 ppm, for BR/EDR: - 20 ppm to 20 ppm , Unit: ppm

get_lrange() → List[int]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:STERror:NMODe:LENergy:LRANge
value: List[int] = driver.configure.rfSettings.dtx.stab.stError.nmode.lowEnergy.
↪get_lrange()
```

Return the symbol timing error under the periodic change according to the test specification for Bluetooth wireless technology (10 values for BR and LE, 3 values for EDR) .

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available. For dirty transmitter parameters according to the test specification for Bluetooth wireless technology, see also 'Dirty Tx Mode'.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return symbol_time_err: decimal Range: - 50 ppm to 50 ppm, for BR/EDR: - 20 ppm to 20 ppm , Unit: ppm

7.1.11.1.1.18 Tmode

class Tmode

Tmode commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.11.1.1.19 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:STERror:TMODe:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:STERror:TMODe:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:STERror:TMODe:LENergy:LE1M
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → List[int]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:STERror:TMODe:LENergy:LE1M
value: List[int] = driver.configure.rfSettings.dtx.stab.stError.tmode.lowEnergy.
↪get_le_1_m()
```

Return the symbol timing error under the periodic change according to the test specification for Bluetooth wireless technology (10 values for BR and LE, 3 values for EDR) .

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available. For dirty transmitter parameters according to the test specification for Bluetooth wireless technology, see also 'Dirty Tx Mode'.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return symbol_time_err: decimal Range: - 50 ppm to 50 ppm, for BR/EDR: - 20 ppm to 20 ppm , Unit: ppm

get_le_2_m() → List[int]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:STERror:TMODe:LENergy:LE2M
value: List[int] = driver.configure.rfSettings.dtx.stab.stError.tmode.lowEnergy.
↪get_le_2_m()
```

Return the symbol timing error under the periodic change according to the test specification for Bluetooth wireless technology (10 values for BR and LE, 3 values for EDR) .

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available. For dirty transmitter parameters according to the test specification for Bluetooth wireless technology, see also 'Dirty Tx Mode'.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return symbol_time_err: decimal Range: - 50 ppm to 50 ppm, for BR/EDR: - 20 ppm to 20 ppm , Unit: ppm

get_lrange() → List[int]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:STERror:TMODe:LENergy:LRANge
value: List[int] = driver.configure.rfSettings.dtx.stab.stError.tmode.lowEnergy.
↪get_lrange()
```

Return the symbol timing error under the periodic change according to the test specification for Bluetooth wireless technology (10 values for BR and LE, 3 values for EDR) .

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available. For dirty transmitter parameters according to the test specification for Bluetooth wireless technology, see also 'Dirty Tx Mode'.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return symbol_time_err: decimal Range: - 50 ppm to 50 ppm, for BR/EDR: - 20 ppm to 20 ppm , Unit: ppm

7.1.11.1.1.20 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:STERror:LENergy:LE1M
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:STERror:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:STERror:LENergy:LE2M
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → List[int]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:STERror:LENergy[:LE1M]
value: List[int] = driver.configure.rfSettings.dtx.stab.stError.lowEnergy.get_
↪le_1_m()
```

Return the symbol timing error under the periodic change according to the test specification for Bluetooth wireless technology (10 values for BR and LE, 3 values for EDR) .

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available. For dirty transmitter parameters according to the test specification for Bluetooth wireless technology, see also 'Dirty Tx Mode'.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return symbol_time_err: decimal Range: - 50 ppm to 50 ppm, for BR/EDR: - 20 ppm to 20 ppm , Unit: ppm

get_le_2_m() → List[int]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:STERror:LENergy:LE2M
value: List[int] = driver.configure.rfSettings.dtx.stab.stError.lowEnergy.get_
↪le_2_m()
```

Return the symbol timing error under the periodic change according to the test specification for Bluetooth wireless technology (10 values for BR and LE, 3 values for EDR) .

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available. For dirty transmitter parameters according to the test specification for Bluetooth wireless technology, see also 'Dirty Tx Mode'.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return symbol_time_err: decimal Range: - 50 ppm to 50 ppm, for BR/EDR: - 20 ppm to 20 ppm , Unit: ppm

get_lrange() → List[int]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:STERror:LENergy:LRANge
value: List[int] = driver.configure.rfSettings.dtx.stab.stError.lowEnergy.get_
↪lrange()
```

Return the symbol timing error under the periodic change according to the test specification for Bluetooth wireless technology (10 values for BR and LE, 3 values for EDR) .

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available. For dirty transmitter parameters according to the test specification for Bluetooth wireless technology, see also 'Dirty Tx Mode'.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return symbol_time_err: decimal Range: - 50 ppm to 50 ppm, for BR/EDR: - 20 ppm to 20 ppm , Unit: ppm

7.1.11.1.1.21 Fdrift

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:FDRift:EDRate
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:FDRift:BRATe
```

class Fdrift

Fdrift commands group definition. 11 total commands, 3 Sub-groups, 2 group commands

get_brate() → List[bool]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:FDRift:BRATe
value: List[bool] = driver.configure.rfSettings.dtx.stab.fdrift.get_brate()
```

Query the dirty transmitter frequency drift set according to the test specification for Bluetooth wireless technology.

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_drift: float Frequency drift is always enabled, according to the test specification for Bluetooth wireless technology.

get_edrate() → List[bool]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:FDRift:EDRate
value: List[bool] = driver.configure.rfSettings.dtx.stab.fdrift.get_edrate()
```

Query the dirty transmitter frequency drift set according to the test specification for Bluetooth wireless technology.

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_drift: float Frequency drift is always enabled, according to the test specification for Bluetooth wireless technology.

Cloning the Group

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

Subgroups

7.1.11.1.1.22 Nmode

class Nmode

Nmode commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.11.1.1.23 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:FDRift:NMODe:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:FDRift:NMODe:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:FDRift:NMODe:LENergy:LE1M
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → List[float]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:FDRift:NMODe:LENergy:LE1M
value: List[float] = driver.configure.rfSettings.dtx.stab.fdrift.nmode.
↪lowEnergy.get_le_1_m()
```

Query the dirty transmitter frequency drift set according to the test specification for Bluetooth wireless technology.

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_drift: float Frequency drift is always enabled, according to the test specification for Bluetooth wireless technology.

get_le_2_m() → List[float]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:FDRift:NMODe:LENergy:LE2M
value: List[float] = driver.configure.rfSettings.dtx.stab.fdrift.nmode.
↪lowEnergy.get_le_2_m()
```

Query the dirty transmitter frequency drift set according to the test specification for Bluetooth wireless technology.

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_drift: float Frequency drift is always enabled, according to the test specification for Bluetooth wireless technology.

get_lrange() → List[float]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:FDRift:NMODe:LENergy:LRANge
value: List[float] = driver.configure.rfSettings.dtx.stab.fdrift.nmode.
↪lowEnergy.get_lrange()
```

Query the dirty transmitter frequency drift set according to the test specification for Bluetooth wireless technology.

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_drift: float Frequency drift is always enabled, according to the test specification for Bluetooth wireless technology.

7.1.11.1.1.24 Tmode

class Tmode

Tmode commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.11.1.1.25 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:FDRift:TMODe:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:FDRift:TMODe:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:FDRift:TMODe:LENergy:LE1M
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → List[float]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:FDRift:TMODe:LENergy:LE1M
value: List[float] = driver.configure.rfSettings.dtx.stab.fdrift.tmode.
↪lowEnergy.get_le_1_m()
```

Query the dirty transmitter frequency drift set according to the test specification for Bluetooth wireless technology.

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_drift: float Frequency drift is always enabled, according to the test specification for Bluetooth wireless technology.

get_le_2_m() → List[float]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:FDRift:TMODe:LENergy:LE2M
value: List[float] = driver.configure.rfSettings.dtx.stab.fdrift.tmode.
↪lowEnergy.get_le_2_m()
```

Query the dirty transmitter frequency drift set according to the test specification for Bluetooth wireless technology.

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_drift: float Frequency drift is always enabled, according to the test specification for Bluetooth wireless technology.

get_lrange() → List[float]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:FDRift:TMODe:LENergy:LRANge
value: List[float] = driver.configure.rfSettings.dtx.stab.fdrift.tmode.
↪lowEnergy.get_lrange()
```

Query the dirty transmitter frequency drift set according to the test specification for Bluetooth wireless technology.

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_drift: float Frequency drift is always enabled, according to the test specification for Bluetooth wireless technology.

7.1.11.1.1.26 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:FDRift:LENergy:LE1M
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:FDRift:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:FDRift:LENergy:LE2M
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → List[float]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:FDRift:LENergy[:LE1M]
value: List[float] = driver.configure.rfSettings.dtx.stab.fdrift.lowEnergy.get_
↪le_1_m()
```

Query the dirty transmitter frequency drift set according to the test specification for Bluetooth wireless technology.

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_drift: float Frequency drift is always enabled, according to the test specification for Bluetooth wireless technology.

get_le_2_m() → List[float]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:FDRift:LENergy:LE2M
value: List[float] = driver.configure.rfSettings.dtx.stab.fdrift.lowEnergy.get_
↪le_2_m()
```

Query the dirty transmitter frequency drift set according to the test specification for Bluetooth wireless technology.

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_drift: float Frequency drift is always enabled, according to the test specification for Bluetooth wireless technology.

get_lrange() → List[float]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:FDRift:LENergy:LRANge
value: List[float] = driver.configure.rfSettings.dtx.stab.fdrift.lowEnergy.get_
↪lrange()
```

Query the dirty transmitter frequency drift set according to the test specification for Bluetooth wireless technology.

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_drift: float Frequency drift is always enabled, according to the test specification for Bluetooth wireless technology.

7.1.11.1.1.27 Foffset

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:FOFFset:EDRate
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:FOFFset:BRATe
```

class Foffset

Foffset commands group definition. 11 total commands, 3 Sub-groups, 2 group commands

get_brate() → List[int]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:FOFFset:BRATe
value: List[int or bool] = driver.configure.rfSettings.dtx.stab.foffset.get_
↪brate()
```

Return the frequency offset under the periodic change according to the test specification for Bluetooth wireless technology (10 values for BR and LE, 3 values for EDR) .

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_offset: decimal | ON | OFF Range: -250 kHz to 250 kHz, Unit: Hz

get_edrate() → List[int]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:FOFFset:EDRate
value: List[int or bool] = driver.configure.rfSettings.dtx.stab.foffset.get_
↪edrate()
```

Return the frequency offset under the periodic change according to the test specification for Bluetooth wireless technology (10 values for BR and LE, 3 values for EDR) .

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_offset: decimal | ON | OFF Range: -250 kHz to 250 kHz, Unit: Hz

Cloning the Group

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

Subgroups

7.1.11.1.1.28 Nmode

class Nmode

Nmode commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.11.1.1.29 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:FOFFset:NMODe:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:FOFFset:NMODe:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:FOFFset:NMODe:LENergy:LE1M
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → List[int]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:FOFFset:NMODe:LENergy:LE1M
value: List[int or bool] = driver.configure.rfSettings.dtx.stab.foffset.nmode.
↪lowEnergy.get_le_1_m()
```

Return the frequency offset under the periodic change according to the test specification for Bluetooth wireless technology (10 values for BR and LE, 3 values for EDR) .

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_offset: decimal | ON | OFF Range: -250 kHz to 250 kHz, Unit: Hz

get_le_2_m() → List[int]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:FOFFset:NMODe:LENergy:LE2M
value: List[int or bool] = driver.configure.rfSettings.dtx.stab.foffset.nmode.
↪lowEnergy.get_le_2_m()
```

Return the frequency offset under the periodic change according to the test specification for Bluetooth wireless technology (10 values for BR and LE, 3 values for EDR) .

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_offset: decimal | ON | OFF Range: -250 kHz to 250 kHz, Unit: Hz

get_lrange() → List[int]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:FOFFset:NMODe:LENergy:LRANge
value: List[int or bool] = driver.configure.rfSettings.dtx.stab.foffset.nmode.
↪lowEnergy.get_lrange()
```

Return the frequency offset under the periodic change according to the test specification for Bluetooth wireless technology (10 values for BR and LE, 3 values for EDR) .

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_offset: decimal | ON | OFF Range: -250 kHz to 250 kHz, Unit: Hz

7.1.11.1.1.30 Tmode

class Tmode

Tmode commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.11.1.1.31 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:FOFFset:TMODe:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:FOFFset:TMODe:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:FOFFset:TMODe:LENergy:LE1M
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → List[int]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:FOFFset:TMODe:LENergy:LE1M
value: List[int or bool] = driver.configure.rfSettings.dtx.stab.foffset.tmode.
↪lowEnergy.get_le_1_m()
```

Return the frequency offset under the periodic change according to the test specification for Bluetooth wireless technology (10 values for BR and LE, 3 values for EDR) .

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_offset: decimal | ON | OFF Range: -250 kHz to 250 kHz, Unit: Hz

get_le_2_m() → List[int]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:FOFFset:TMODe:LENergy:LE2M
value: List[int or bool] = driver.configure.rfSettings.dtx.stab.foffset.tmode.
↪lowEnergy.get_le_2_m()
```

Return the frequency offset under the periodic change according to the test specification for Bluetooth wireless technology (10 values for BR and LE, 3 values for EDR) .

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_offset: decimal | ON | OFF Range: -250 kHz to 250 kHz, Unit: Hz

get_lrange() → List[int]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:FOFFset:TMODe:LENergy:LRANge
value: List[int or bool] = driver.configure.rfSettings.dtx.stab.foffset.tmode.
↪lowEnergy.get_lrange()
```

Return the frequency offset under the periodic change according to the test specification for Bluetooth wireless technology (10 values for BR and LE, 3 values for EDR) .

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_offset: decimal | ON | OFF Range: -250 kHz to 250 kHz, Unit: Hz

7.1.11.1.1.32 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:FOFFset:LENergy:LE1M
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:FOFFset:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STAB:FOFFset:LENergy:LE2M
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → List[int]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:FOFFset:LENergy[:LE1M]
value: List[int or bool] = driver.configure.rfSettings.dtx.stab.foffset.
↪lowEnergy.get_le_1_m()
```

Return the frequency offset under the periodic change according to the test specification for Bluetooth wireless technology (10 values for BR and LE, 3 values for EDR) .

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_offset: decimal | ON | OFF Range: -250 kHz to 250 kHz, Unit: Hz

get_le_2_m() → List[int]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:FOFFset:LENergy:LE2M
value: List[int or bool] = driver.configure.rfSettings.dtx.stab.foffset.
↪lowEnergy.get_le_2_m()
```

Return the frequency offset under the periodic change according to the test specification for Bluetooth wireless technology (10 values for BR and LE, 3 values for EDR) .

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_offset: decimal | ON | OFF Range: -250 kHz to 250 kHz, Unit: Hz

get_lrange() → List[int]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:STAB:FOFFset:LENergy:LRANge
value: List[int or bool] = driver.configure.rfSettings.dtx.stab.foffset.
↪lowEnergy.get_lrange()
```

Return the frequency offset under the periodic change according to the test specification for Bluetooth wireless technology (10 values for BR and LE, 3 values for EDR) .

INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_offset: decimal | ON | OFF Range: -250 kHz to 250 kHz, Unit: Hz

7.1.11.1.2 Sing

class Sing
Sing commands group definition. 51 total commands, 4 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.11.1.2.1 Mindex

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:MINDex:BRATe
```

class Mindex
Mindex commands group definition. 18 total commands, 5 Sub-groups, 1 group commands

get_brate() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:MINDex:BRATe
value: float or bool = driver.configure.rfSettings.dtx.sing.mindex.get_brate()
```

Specifies the modulation corruption of the signal. Modulation index of 0.32 means no corruption.

return mod_index: numeric | ON | OFF Range: 0.2 to 0.44 Additional ON/OFF enables/disables modulation index.

set_brate(*mod_index: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:MINDex:BRATe
driver.configure.rfSettings.dtx.sing.mindex.set_brate(mod_index = 1.0)
```

Specifies the modulation corruption of the signal. Modulation index of 0.32 means no corruption.

param mod_index numeric | ON | OFF Range: 0.2 to 0.44 Additional ON/OFF enables/disables modulation index.

Cloning the Group

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

Subgroups

7.1.11.1.2.2 Nmode

class Nmode
Nmode commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.11.1.2.3 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:MINDex:NMODe:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:MINDex:NMODe:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:MINDex:NMODe:LENergy:LE1M
```

class LowEnergy
LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:MINDex:NMODe:LENergy:LE1M
value: float or bool = driver.configure.rfSettings.dtx.sing.mindex.nmode.
↪lowEnergy.get_le_1_m()
```

Specifies the modulation corruption of the signal. Modulation index of 0.5 means no corruption.
INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return mod_index: numeric | ON | OFF Range: 0.4 to 0.6 Additional ON/OFF enables/disables modulation index.

get_le_2_m() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:MINDex:NMODe:LENergy:LE2M
value: float or bool = driver.configure.rfSettings.dtx.sing.mindex.nmode.
↪lowEnergy.get_le_2_m()
```

Specifies the modulation corruption of the signal. Modulation index of 0.5 means no corruption.

INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return mod_index: numeric | ON | OFF Range: 0.4 to 0.6 Additional ON/OFF enables/disables modulation index.

get_lrange() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:MINDex:NMODe:LENergy:LRANge
value: float or bool = driver.configure.rfSettings.dtx.sing.mindex.nmode.
↪lowEnergy.get_lrange()
```

Specifies the modulation corruption of the signal. Modulation index of 0.5 means no corruption.

INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return mod_index: numeric | ON | OFF Range: 0.4 to 0.6 Additional ON/OFF enables/disables modulation index.

set_le_1_m(*mod_index: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:MINDex:NMODe:LENergy:LE1M
driver.configure.rfSettings.dtx.sing.mindex.nmode.lowEnergy.set_le_1_m(mod_
↪index = 1.0)
```

Specifies the modulation corruption of the signal. Modulation index of 0.5 means no corruption.

INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param mod_index numeric | ON | OFF Range: 0.4 to 0.6 Additional ON/OFF enables/disables modulation index.

set_le_2_m(*mod_index: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:MINDex:NMODe:LENergy:LE2M
driver.configure.rfSettings.dtx.sing.mindex.nmode.lowEnergy.set_le_2_m(mod_
↪index = 1.0)
```

Specifies the modulation corruption of the signal. Modulation index of 0.5 means no corruption.

INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param mod_index numeric | ON | OFF Range: 0.4 to 0.6 Additional ON/OFF enables/disables modulation index.

set_lrange(*mod_index: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:MINDex:NMODe:LENergy:LRANge
driver.configure.rfSettings.dtx.sing.mindex.nmode.lowEnergy.set_lrange(mod_
↪index = 1.0)
```

Specifies the modulation corruption of the signal. Modulation index of 0.5 means no corruption.

INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param mod_index numeric | ON | OFF Range: 0.4 to 0.6 Additional ON/OFF enables/disables modulation index.

7.1.11.1.2.4 Standard

class Standard
Standard commands group definition. 6 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.11.1.2.5 Tmode

class Tmode
Tmode commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.rfSettings.dtx.sing.mindex.standard.tmode.clone()
```

Subgroups

7.1.11.1.2.6 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:MINDex:STANdard:TMODe:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:MINDex:STANdard:TMODe:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:MINDex:STANdard:TMODe:LENergy:LE1M
```

class LowEnergy
LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:MINDex:STANdard:TMODe:LENergy:LE1M
value: float or bool = driver.configure.rfSettings.dtx.sing.mindex.standard.
↪tmode.lowEnergy.get_le_1_m()
```

Specifies the modulation corruption of the signal. Modulation index of 0.5 means no corruption.
INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return mod_index: numeric | ON | OFF Range: 0.4 to 0.6 Additional ON/OFF enables/disables modulation index.

get_le_2_m() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:MINDex:STANdard:TMODe:LENergy:LE2M
value: float or bool = driver.configure.rfSettings.dtx.sing.mindex.standard.
↪tmode.lowEnergy.get_le_2_m()
```

Specifies the modulation corruption of the signal. Modulation index of 0.5 means no corruption.

INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return mod_index: numeric | ON | OFF Range: 0.4 to 0.6 Additional ON/OFF enables/disables modulation index.

get_lrange() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:MINDex:STANdard:TMODe:LENergy:LRANge
value: float or bool = driver.configure.rfSettings.dtx.sing.mindex.standard.
↪tmode.lowEnergy.get_lrange()
```

Specifies the modulation corruption of the signal. Modulation index of 0.5 means no corruption.

INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return mod_index: numeric | ON | OFF Range: 0.4 to 0.6 Additional ON/OFF enables/disables modulation index.

set_le_1_m(*mod_index: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:MINDex:STANdard:TMODe:LENergy:LE1M
driver.configure.rfSettings.dtx.sing.mindex.standard.tmode.lowEnergy.set_le_1_
↪m(mod_index = 1.0)
```

Specifies the modulation corruption of the signal. Modulation index of 0.5 means no corruption.

INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param mod_index numeric | ON | OFF Range: 0.4 to 0.6 Additional ON/OFF enables/disables modulation index.

set_le_2_m(*mod_index: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:MINDex:STANdard:TMODe:LENergy:LE2M
driver.configure.rfSettings.dtx.sing.mindex.standard.tmode.lowEnergy.set_le_2_
↪m(mod_index = 1.0)
```

Specifies the modulation corruption of the signal. Modulation index of 0.5 means no corruption.

INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param mod_index numeric | ON | OFF Range: 0.4 to 0.6 Additional ON/OFF enables/disables modulation index.

set_lrange(*mod_index: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:MINDex:STANdard:TMODe:LENergy:LRANge
driver.configure.rfSettings.dtx.sing.mindex.standard.tmode.lowEnergy.set_
↪lrange(mod_index = 1.0)
```

Specifies the modulation corruption of the signal. Modulation index of 0.5 means no corruption.

INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param mod_index numeric | ON | OFF Range: 0.4 to 0.6 Additional ON/OFF enables/disables modulation index.

7.1.11.1.2.7 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:MINDex:STANdard:LENergy:LE1M
CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:MINDex:STANdard:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:MINDex:STANdard:LENergy:LE2M
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:MINDex:STANdard:LENergy[:LE1M]
value: float or bool = driver.configure.rfSettings.dtx.sing.mindex.standard.
↪lowEnergy.get_le_1_m()
```

Specifies the modulation corruption of the signal. Modulation index of 0.5 means no corruption.

INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return mod_index: numeric | ON | OFF Range: 0.4 to 0.6 Additional ON/OFF enables/disables modulation index.

get_le_2_m() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:MINDex:STANdard:LENergy:LE2M
value: float or bool = driver.configure.rfSettings.dtx.sing.mindex.standard.
↪lowEnergy.get_le_2_m()
```

Specifies the modulation corruption of the signal. Modulation index of 0.5 means no corruption.

INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return mod_index: numeric | ON | OFF Range: 0.4 to 0.6 Additional ON/OFF enables/disables modulation index.

get_lrange() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:MINDex:STANdard:LENergy:LRANge
value: float or bool = driver.configure.rfSettings.dtx.sing.mindex.standard.
↪lowEnergy.get_lrange()
```

Specifies the modulation corruption of the signal. Modulation index of 0.5 means no corruption.

INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return mod_index: numeric | ON | OFF Range: 0.4 to 0.6 Additional ON/OFF enables/disables modulation index.

set_le_1_m(*mod_index: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:MINDex:STANdard:LENergy[:LE1M]
driver.configure.rfSettings.dtx.sing.mindex.standard.lowEnergy.set_le_1_m(mod_
↪index = 1.0)
```

Specifies the modulation corruption of the signal. Modulation index of 0.5 means no corruption.

INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param mod_index numeric | ON | OFF Range: 0.4 to 0.6 Additional ON/OFF enables/disables modulation index.

set_le_2_m(*mod_index: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:MINDex:STANdard:LENergy:LE2M
driver.configure.rfSettings.dtx.sing.mindex.standard.lowEnergy.set_le_2_m(mod_
↪index = 1.0)
```

Specifies the modulation corruption of the signal. Modulation index of 0.5 means no corruption.

INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param mod_index numeric | ON | OFF Range: 0.4 to 0.6 Additional ON/OFF enables/disables modulation index.

`set_lrange`(*mod_index: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:MINDex:STANdard:LENergy:LRANge
driver.configure.rfSettings.dtx.sing.mindex.standard.lowEnergy.set_lrange(mod_
↪index = 1.0)
```

Specifies the modulation corruption of the signal. Modulation index of 0.5 means no corruption.

INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param mod_index numeric | ON | OFF Range: 0.4 to 0.6 Additional ON/OFF enables/disables modulation index.

7.1.11.1.2.8 Stable

`class Stable`

Stable commands group definition. 6 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.11.1.2.9 Tmode

class Tmode
Tmode commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.rfSettings.dtx.sing.mindex.stable.tmode.clone()
```

Subgroups

7.1.11.1.2.10 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:MINDex:STABle:TMODe:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:MINDex:STABle:TMODe:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:MINDex:STABle:TMODe:LENergy:LE1M
```

class LowEnergy
LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:MINDex:STABle:TMODe:LENergy:LE1M
value: float or bool = driver.configure.rfSettings.dtx.sing.mindex.stable.tmode.
↪lowEnergy.get_le_1_m()
```

Specifies the modulation corruption of the signal. Modulation index of 0.5 means no corruption.
INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode:

Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are a
INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return mod_index: numeric | ON | OFF Range: 0.4 to 0.6 Additional ON/OFF enables/disables modulation index.

get_le_2_m() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:MINDex:STABle:TMODe:LENergy:LE2M
value: float or bool = driver.configure.rfSettings.dtx.sing.mindex.stable.tmode.
↪lowEnergy.get_le_2_m()
```

Specifies the modulation corruption of the signal. Modulation index of 0.5 means no corruption.
INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode:

Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are a
INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return mod_index: numeric | ON | OFF Range: 0.4 to 0.6 Additional ON/OFF enables/disables modulation index.

get_lrange() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:MINDex:STABle:TMODe:LENergy:LRANge
value: float or bool = driver.configure.rfSettings.dtx.sing.mindex.stable.tmode.
↪lowEnergy.get_lrange()
```

Specifies the modulation corruption of the signal. Modulation index of 0.5 means no corruption.
INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode:

Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are a
INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return mod_index: numeric | ON | OFF Range: 0.4 to 0.6 Additional ON/OFF enables/disables modulation index.

set_le_1_m(*mod_index: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:MINDex:STABle:TMODe:LENergy:LE1M
driver.configure.rfSettings.dtx.sing.mindex.stable.tmode.lowEnergy.set_le_1_
↪m(mod_index = 1.0)
```

Specifies the modulation corruption of the signal. Modulation index of 0.5 means no corruption.
INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode:

Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are a
INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param mod_index numeric | ON | OFF Range: 0.4 to 0.6 Additional ON/OFF enables/disables modulation index.

`set_le_2_m`(*mod_index: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:MINDex:STABle:TMODe:LENergy:LE2M
driver.configure.rfSettings.dtx.sing.mindex.stable.tmode.lowEnergy.set_le_2_
↪m(mod_index = 1.0)
```

Specifies the modulation corruption of the signal. Modulation index of 0.5 means no corruption.
INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode:

Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are a
INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param mod_index numeric | ON | OFF Range: 0.4 to 0.6 Additional ON/OFF enables/disables modulation index.

`set_lrange`(*mod_index: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:MINDex:STABle:TMODe:LENergy:LRANge
driver.configure.rfSettings.dtx.sing.mindex.stable.tmode.lowEnergy.set_
↪lrange(mod_index = 1.0)
```

Specifies the modulation corruption of the signal. Modulation index of 0.5 means no corruption.
INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode:

Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are a
INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param mod_index numeric | ON | OFF Range: 0.4 to 0.6 Additional ON/OFF enables/disables modulation index.

7.1.11.1.2.11 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:MINDex:STABle:LENergy:LE1M
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:MINDex:STABle:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:MINDex:STABle:LENergy:LE2M
```

class LowEnergy
LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:MINDex:STABle:LENergy[:LE1M]
value: float or bool = driver.configure.rfSettings.dtx.sing.mindex.stable.
↪lowEnergy.get_le_1_m()
```

Specifies the modulation corruption of the signal. Modulation index of 0.5 means no corruption.
INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode:

Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are a
INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return mod_index: numeric | ON | OFF Range: 0.4 to 0.6 Additional ON/OFF enables/disables modulation index.

get_le_2_m() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:MINDex:STABle:LENergy:LE2M
value: float or bool = driver.configure.rfSettings.dtx.sing.mindex.stable.
↪lowEnergy.get_le_2_m()
```

Specifies the modulation corruption of the signal. Modulation index of 0.5 means no corruption.
INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode:

Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are a
INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return mod_index: numeric | ON | OFF Range: 0.4 to 0.6 Additional ON/OFF enables/disables modulation index.

`get_lrange`() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:MINDex:STABle:LENergy:LRANge
value: float or bool = driver.configure.rfSettings.dtx.sing.mindex.stable.
↪lowEnergy.get_lrange()
```

Specifies the modulation corruption of the signal. Modulation index of 0.5 means no corruption.
INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode:

Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are a
INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return mod_index: numeric | ON | OFF Range: 0.4 to 0.6 Additional ON/OFF enables/disables modulation index.

`set_le_1_m`(*mod_index: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:MINDex:STABle:LENergy[:LE1M]
driver.configure.rfSettings.dtx.sing.mindex.stable.lowEnergy.set_le_1_m(mod_
↪index = 1.0)
```

Specifies the modulation corruption of the signal. Modulation index of 0.5 means no corruption.
INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode:

Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are a
INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param mod_index numeric | ON | OFF Range: 0.4 to 0.6 Additional ON/OFF enables/disables modulation index.

set_le_2_m(*mod_index: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:MINDex:STABle:LENergy:LE2M
driver.configure.rfSettings.dtx.sing.mindex.stable.lowEnergy.set_le_2_m(mod_
↪index = 1.0)
```

Specifies the modulation corruption of the signal. Modulation index of 0.5 means no corruption.
INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode:

Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are a
INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param mod_index numeric | ON | OFF Range: 0.4 to 0.6 Additional ON/OFF enables/disables modulation index.

set_lrange(*mod_index: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:MINDex:STABle:LENergy:LRANge
driver.configure.rfSettings.dtx.sing.mindex.stable.lowEnergy.set_lrange(mod_
↪index = 1.0)
```

Specifies the modulation corruption of the signal. Modulation index of 0.5 means no corruption.
INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode:

Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are a
INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param mod_index numeric | ON | OFF Range: 0.4 to 0.6 Additional ON/OFF enables/disables modulation index.

7.1.11.1.2.12 Tmode

class Tmode

Tmode commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.11.1.2.13 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:MINDex:TMODe:LENergy:LE1M
```

class LowEnergy

LowEnergy commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_le_1_m() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:MINDex:TMODe:LENergy:LE1M
value: float or bool = driver.configure.rfSettings.dtx.sing.mindex.tmode.
↪lowEnergy.get_le_1_m()
```

No command help available

return mod_index: No help available

set_le_1_m(*mod_index: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:MINDex:TMODe:LENergy:LE1M
driver.configure.rfSettings.dtx.sing.mindex.tmode.lowEnergy.set_le_1_m(mod_
↪index = 1.0)
```

No command help available

param mod_index No help available

7.1.11.1.2.14 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:MINDex:LENergy:LE1M
```

class LowEnergy

LowEnergy commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_le_1_m() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:MINDex:LENergy[:LE1M]
value: float or bool = driver.configure.rfSettings.dtx.sing.mindex.lowEnergy.
↪get_le_1_m()
```

No command help available

return mod_index: No help available

set_le_1_m(*mod_index: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:MINDex:LENergy[:LE1M]
driver.configure.rfSettings.dtx.sing.mindex.lowEnergy.set_le_1_m(mod_index = 1.
↪0)
```

No command help available

param mod_index No help available

7.1.11.1.2.15 StError

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:STERror:EDRate
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:STERror:BRATe
```

class StError

StError commands group definition. 11 total commands, 3 Sub-groups, 2 group commands

get_brate() → RsCmwBluetoothSig.enums.SymbolTimeError

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:STERror:BRATe
value: enums.SymbolTimeError = driver.configure.rfSettings.dtx.sing.stError.get_
↪brate()
```

Specify the symbol timing error of the signal.

return symbol_time_err: OFF | NEG20 | POS20 No symbol timing error, - 20 ppm or 20 ppm

get_edrate() → RsCmwBluetoothSig.enums.SymbolTimeError

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:STERror:EDRate
value: enums.SymbolTimeError = driver.configure.rfSettings.dtx.sing.stError.get_
↪edrate()
```

Specify the symbol timing error of the signal.

return symbol_time_err: OFF | NEG20 | POS20 No symbol timing error, - 20 ppm or 20 ppm

set_brate(*symbol_time_err: RsCmwBluetoothSig.enums.SymbolTimeError*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:STERror:BRATe
driver.configure.rfSettings.dtx.sing.stError.set_brate(symbol_time_err = enums.
↪SymbolTimeError.NEG20)
```

Specify the symbol timing error of the signal.

param symbol_time_err OFF | NEG20 | POS20 No symbol timing error, - 20 ppm or 20 ppm

set_edrate(*symbol_time_err: RsCmwBluetoothSig.enums.SymbolTimeError*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:STERror:EDRate
driver.configure.rfSettings.dtx.sing.stError.set_edrate(symbol_time_err = enums.
↪SymbolTimeError.NEG20)
```

Specify the symbol timing error of the signal.

param symbol_time_err OFF | NEG20 | POS20 No symbol timing error, - 20 ppm or 20 ppm

Cloning the Group

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

Subgroups

7.1.11.1.2.16 Nmode

class Nmode

Nmode commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.11.1.2.17 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:STERror:NMODe:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:STERror:NMODe:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:STERror:NMODe:LENergy:LE1M
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → RsCmwBluetoothSig.enums.SymbolTimeErrorLe

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:STERror:NMODe:LENergy:LE1M
value: enums.SymbolTimeErrorLe = driver.configure.rfSettings.dtx.sing.stError.
↪nmode.lowEnergy.get_le_1_m()
```

Specifies the symbol timing error of the LE signal. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return symbol_time_err: OFF | NEG50 | POS50 No symbol timing error, - 50 ppm or 50 ppm

get_le_2_m() → RsCmwBluetoothSig.enums.SymbolTimeErrorLe

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:STERror:NMODe:LENergy:LE2M
value: enums.SymbolTimeErrorLe = driver.configure.rfSettings.dtx.sing.stError.
↪nmode.lowEnergy.get_le_2_m()
```

Specifies the symbol timing error of the LE signal. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return symbol_time_err: OFF | NEG50 | POS50 No symbol timing error, - 50 ppm or 50 ppm

get_lrange() → RsCmwBluetoothSig.enums.SymbolTimeErrorLe

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:STERror:NMODe:LENergy:LRANge
value: enums.SymbolTimeErrorLe = driver.configure.rfSettings.dtx.sing.stError.
↪nmode.lowEnergy.get_lrange()
```

Specifies the symbol timing error of the LE signal. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return symbol_time_err: OFF | NEG50 | POS50 No symbol timing error, - 50 ppm or 50 ppm

set_le_1_m(*symbol_time_err: RsCmwBluetoothSig.enums.SymbolTimeErrorLe*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:STERror:NMODe:LENergy:LE1M
driver.configure.rfSettings.dtx.sing.stError.nmode.lowEnergy.set_le_1_m(symbol_
↪time_err = enums.SymbolTimeErrorLe.NEG50)
```

Specifies the symbol timing error of the LE signal. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param symbol_time_err OFF | NEG50 | POS50 No symbol timing error, - 50 ppm or 50 ppm

set_le_2_m(*symbol_time_err: RsCmwBluetoothSig.enums.SymbolTimeErrorLe*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:STERror:NMODe:LENergy:LE2M
driver.configure.rfSettings.dtx.sing.stError.nmode.lowEnergy.set_le_2_m(symbol_
↪time_err = enums.SymbolTimeErrorLe.NEG50)
```

Specifies the symbol timing error of the LE signal. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param symbol_time_err OFF | NEG50 | POS50 No symbol timing error, - 50 ppm or 50 ppm

`set_lrange`(*symbol_time_err: RsCmwBluetoothSig.enums.SymbolTimeErrorLe*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:STERror:NMODe:LENergy:LRANge
driver.configure.rfSettings.dtx.sing.stError.nmode.lowEnergy.set_lrange(symbol_
↪time_err = enums.SymbolTimeErrorLe.NEG50)
```

Specifies the symbol timing error of the LE signal. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param symbol_time_err OFF | NEG50 | POS50 No symbol timing error, - 50 ppm or 50 ppm

7.1.11.1.2.18 Tmode

`class Tmode`

Tmode commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.11.1.2.19 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:STERror:TMODe:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:STERror:TMODe:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:STERror:TMODe:LENergy:LE1M
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → RsCmwBluetoothSig.enums.SymbolTimeErrorLe

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:STERror:TMODe:LENergy:LE1M
value: enums.SymbolTimeErrorLe = driver.configure.rfSettings.dtx.sing.stError.
↪tmode.lowEnergy.get_le_1_m()
```

Specifies the symbol timing error of the LE signal. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return symbol_time_err: OFF | NEG50 | POS50 No symbol timing error, - 50 ppm or 50 ppm

get_le_2_m() → RsCmwBluetoothSig.enums.SymbolTimeErrorLe

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:STERror:TMODe:LENergy:LE2M
value: enums.SymbolTimeErrorLe = driver.configure.rfSettings.dtx.sing.stError.
↪tmode.lowEnergy.get_le_2_m()
```

Specifies the symbol timing error of the LE signal. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return symbol_time_err: OFF | NEG50 | POS50 No symbol timing error, - 50 ppm or 50 ppm

get_lrange() → RsCmwBluetoothSig.enums.SymbolTimeErrorLe

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:STERror:TMODe:LENergy:LRANge
value: enums.SymbolTimeErrorLe = driver.configure.rfSettings.dtx.sing.stError.
↪tmode.lowEnergy.get_lrange()
```

Specifies the symbol timing error of the LE signal. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return symbol_time_err: OFF | NEG50 | POS50 No symbol timing error, - 50 ppm or 50 ppm

set_le_1_m(*symbol_time_err: RsCmwBluetoothSig.enums.SymbolTimeErrorLe*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:STERror:TMODe:LENergy:LE1M
driver.configure.rfSettings.dtx.sing.stError.tmode.lowEnergy.set_le_1_m(symbol_
↪time_err = enums.SymbolTimeErrorLe.NEG50)
```

Specifies the symbol timing error of the LE signal. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param symbol_time_err OFF | NEG50 | POS50 No symbol timing error, - 50 ppm or 50 ppm

set_le_2_m(*symbol_time_err: RsCmwBluetoothSig.enums.SymbolTimeErrorLe*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:STERror:TMODe:LENergy:LE2M
driver.configure.rfSettings.dtx.sing.stError.tmode.lowEnergy.set_le_2_m(symbol_
↪time_err = enums.SymbolTimeErrorLe.NEG50)
```

Specifies the symbol timing error of the LE signal. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param symbol_time_err OFF | NEG50 | POS50 No symbol timing error, - 50 ppm or 50 ppm

set_lrange(*symbol_time_err: RsCmwBluetoothSig.enums.SymbolTimeErrorLe*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:STERror:TMODe:LENergy:LRANge
driver.configure.rfSettings.dtx.sing.stError.tmode.lowEnergy.set_lrange(symbol_
↪time_err = enums.SymbolTimeErrorLe.NEG50)
```

Specifies the symbol timing error of the LE signal. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param symbol_time_err OFF | NEG50 | POS50 No symbol timing error, - 50 ppm or 50 ppm

7.1.11.1.2.20 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:STERror:LENergy:LE1M
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:STERror:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:STERror:LENergy:LE2M
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → RsCmwBluetoothSig.enums.SymbolTimeErrorLe

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:STERror:LENergy[:LE1M]
value: enums.SymbolTimeErrorLe = driver.configure.rfSettings.dtx.sing.stError.
↪lowEnergy.get_le_1_m()
```

Specifies the symbol timing error of the LE signal. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return symbol_time_err: OFF | NEG50 | POS50 No symbol timing error, - 50 ppm or 50 ppm

get_le_2_m() → RsCmwBluetoothSig.enums.SymbolTimeErrorLe

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:STERror:LENergy:LE2M
value: enums.SymbolTimeErrorLe = driver.configure.rfSettings.dtx.sing.stError.
↪lowEnergy.get_le_2_m()
```

Specifies the symbol timing error of the LE signal. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return symbol_time_err: OFF | NEG50 | POS50 No symbol timing error, - 50 ppm or 50 ppm

get_lrange() → RsCmwBluetoothSig.enums.SymbolTimeErrorLe

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:STERror:LENergy:LRANge
value: enums.SymbolTimeErrorLe = driver.configure.rfSettings.dtx.sing.stError.
↪lowEnergy.get_lrange()
```

Specifies the symbol timing error of the LE signal. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return symbol_time_err: OFF | NEG50 | POS50 No symbol timing error, - 50 ppm or 50 ppm

set_le_1_m(*symbol_time_err: RsCmwBluetoothSig.enums.SymbolTimeErrorLe*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:STERror:LENergy[:LE1M]
driver.configure.rfSettings.dtx.sing.stError.lowEnergy.set_le_1_m(symbol_time_
↪err = enums.SymbolTimeErrorLe.NEG50)
```

Specifies the symbol timing error of the LE signal. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param symbol_time_err OFF | NEG50 | POS50 No symbol timing error, - 50 ppm or 50 ppm

set_le_2_m(*symbol_time_err: RsCmwBluetoothSig.enums.SymbolTimeErrorLe*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:STERror:LENergy:LE2M
driver.configure.rfSettings.dtx.sing.stError.lowEnergy.set_le_2_m(symbol_time_
↪err = enums.SymbolTimeErrorLe.NEG50)
```

Specifies the symbol timing error of the LE signal. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param symbol_time_err OFF | NEG50 | POS50 No symbol timing error, - 50 ppm or 50 ppm

set_lrange(*symbol_time_err: RsCmwBluetoothSig.enums.SymbolTimeErrorLe*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:STERror:LENergy:LRANge
driver.configure.rfSettings.dtx.sing.stError.lowEnergy.set_lrange(symbol_time_
↪err = enums.SymbolTimeErrorLe.NEG50)
```

Specifies the symbol timing error of the LE signal. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param symbol_time_err OFF | NEG50 | POS50 No symbol timing error, - 50 ppm or 50 ppm

7.1.11.1.2.21 Fdrift

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:FDRift:EDRate
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:FDRift:BRATe
```

class Fdrift

Fdrift commands group definition. 11 total commands, 3 Sub-groups, 2 group commands

get_brate() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:FDRift:BRATe
value: bool = driver.configure.rfSettings.dtx.sing.fdrift.get_brate()
```

Enable/disable the periodic change of frequency offset. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_drift: OFF | ON

get_edrate() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:FDRift:EDRate
value: bool = driver.configure.rfSettings.dtx.sing.fdrift.get_edrate()
```

Enable/disable the periodic change of frequency offset. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_drift: OFF | ON

set_brate(*freq_drift: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:FDRift:BRATe
driver.configure.rfSettings.dtx.sing.fdrift.set_brate(freq_drift = False)
```

Enable/disable the periodic change of frequency offset. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param freq_drift OFF | ON

set_edrate(*freq_drift: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:FDRift:EDRate
driver.configure.rfSettings.dtx.sing.fdrift.set_edrate(freq_drift = False)
```

Enable/disable the periodic change of frequency offset. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param freq_drift OFF | ON

Cloning the Group

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

Subgroups

7.1.11.1.2.22 Nmode

class Nmode

Nmode commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.11.1.2.23 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:FDRift:NMODe:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:FDRift:NMODe:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:FDRift:NMODe:LENergy:LE1M
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:FDRift:NMODe:LENergy:LE1M
value: bool = driver.configure.rfSettings.dtx.sing.fdrift.nmode.lowEnergy.get_
↪le_1_m()
```

Enable/disable the periodic change of frequency offset. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_drift: OFF | ON

get_le_2_m() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:FDRift:NMODe:LENergy:LE2M
value: bool = driver.configure.rfSettings.dtx.sing.fdrift.nmode.lowEnergy.get_
↪le_2_m()
```

Enable/disable the periodic change of frequency offset. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_drift: OFF | ON

get_lrange() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:FDRift:NMODe:LENergy:LRANge
value: bool = driver.configure.rfSettings.dtx.sing.fdrift.nmode.lowEnergy.get_
↪lrange()
```

Enable/disable the periodic change of frequency offset. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_drift: OFF | ON

set_le_1_m(*freq_drift: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:FDRift:NMODe:LENergy:LE1M
driver.configure.rfSettings.dtx.sing.fdrift.nmode.lowEnergy.set_le_1_m(freq_
↪drift = False)
```

Enable/disable the periodic change of frequency offset. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param freq_drift OFF | ON

`set_le_2_m`(*freq_drift: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:FDRift:NMODe:LENergy:LE2M
driver.configure.rfSettings.dtx.sing.fdrift.nmode.lowEnergy.set_le_2_m(freq_
↪drift = False)
```

Enable/disable the periodic change of frequency offset. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param freq_drift OFF | ON

`set_lrange`(*freq_drift: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:FDRift:NMODe:LENergy:LRANge
driver.configure.rfSettings.dtx.sing.fdrift.nmode.lowEnergy.set_lrange(freq_
↪drift = False)
```

Enable/disable the periodic change of frequency offset. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param freq_drift OFF | ON

7.1.11.1.2.24 Tmode

class Tmode

Tmode commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.11.1.2.25 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:FDRift:TMODe:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:FDRift:TMODe:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:FDRift:TMODe:LENergy:LE1M
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:FDRift:TMODe:LENergy:LE1M
value: bool = driver.configure.rfSettings.dtx.sing.fdrift.tmode.lowEnergy.get_
↪le_1_m()
```

Enable/disable the periodic change of frequency offset. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_drift: OFF | ON

get_le_2_m() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:FDRift:TMODe:LENergy:LE2M
value: bool = driver.configure.rfSettings.dtx.sing.fdrift.tmode.lowEnergy.get_
↪le_2_m()
```

Enable/disable the periodic change of frequency offset. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_drift: OFF | ON

get_lrange() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:FDRift:TMODe:LENergy:LRANge
value: bool = driver.configure.rfSettings.dtx.sing.fdrift.tmode.lowEnergy.get_
↪lrange()
```

Enable/disable the periodic change of frequency offset. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_drift: OFF | ON

set_le_1_m(*freq_drift: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:FDRift:TMODe:LENergy:LE1M
driver.configure.rfSettings.dtx.sing.fdrift.tmode.lowEnergy.set_le_1_m(freq_
↪drift = False)
```

Enable/disable the periodic change of frequency offset. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param freq_drift OFF | ON

`set_le_2_m`(*freq_drift: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:FDRift:TMODe:LENergy:LE2M
driver.configure.rfSettings.dtx.sing.fdrift.tmode.lowEnergy.set_le_2_m(freq_
↪drift = False)
```

Enable/disable the periodic change of frequency offset. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param freq_drift OFF | ON

`set_lrange`(*freq_drift: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:FDRift:TMODe:LENergy:LRANge
driver.configure.rfSettings.dtx.sing.fdrift.tmode.lowEnergy.set_lrange(freq_
↪drift = False)
```

Enable/disable the periodic change of frequency offset. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param freq_drift OFF | ON

7.1.11.1.2.26 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:FDRift:LENergy:LE1M
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:FDRift:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:FDRift:LENergy:LE2M
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:FDRift:LENergy[:LE1M]
value: bool = driver.configure.rfSettings.dtx.sing.fdrift.lowEnergy.get_le_1_m()
```

Enable/disable the periodic change of frequency offset. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_drift: OFF | ON

get_le_2_m() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:FDRift:LENergy:LE2M
value: bool = driver.configure.rfSettings.dtx.sing.fdrift.lowEnergy.get_le_2_m()
```

Enable/disable the periodic change of frequency offset. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_drift: OFF | ON

get_lrange() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:FDRift:LENergy:LRANge
value: bool = driver.configure.rfSettings.dtx.sing.fdrift.lowEnergy.get_lrange()
```

Enable/disable the periodic change of frequency offset. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_drift: OFF | ON

set_le_1_m(*freq_drift: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:FDRift:LENergy[:LE1M]
driver.configure.rfSettings.dtx.sing.fdrift.lowEnergy.set_le_1_m(freq_drift =␣
↪False)
```

Enable/disable the periodic change of frequency offset. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param freq_drift OFF | ON

set_le_2_m(*freq_drift: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:FDRift:LENergy:LE2M
driver.configure.rfSettings.dtx.sing.fdrift.lowEnergy.set_le_2_m(freq_drift =␣
↪False)
```

Enable/disable the periodic change of frequency offset. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param freq_drift OFF | ON

`set_lrange`(*freq_drift: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:FDRift:LENergy:LRANge
driver.configure.rfSettings.dtx.sing.fdrift.lowEnergy.set_lrange(freq_drift =␣
↪False)
```

Enable/disable the periodic change of frequency offset. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param freq_drift OFF | ON

7.1.11.1.2.27 Foffset

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:FOFFset:EDRate
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:FOFFset:BRATe
```

`class Foffset`

Foffset commands group definition. 11 total commands, 3 Sub-groups, 2 group commands

`get_brate`() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:FOFFset:BRATe
value: int or bool = driver.configure.rfSettings.dtx.sing.foffset.get_brate()
```

Specify the constant frequency offset to be added to the center frequency. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_offset: numeric | ON | OFF Range: -250 kHz to 250 kHz Additional ON/OFF enables/disables constant frequency offset

get_edrate() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:FOFFset:EDRate
value: int or bool = driver.configure.rfSettings.dtx.sing.foffset.get_edrate()
```

Specify the constant frequency offset to be added to the center frequency. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_offset: numeric | ON | OFF Range: -250 kHz to 250 kHz Additional ON/OFF enables/disables constant frequency offset

set_brate(*freq_offset: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:FOFFset:BRATe
driver.configure.rfSettings.dtx.sing.foffset.set_brate(freq_offset = 1)
```

Specify the constant frequency offset to be added to the center frequency. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param freq_offset numeric | ON | OFF Range: -250 kHz to 250 kHz Additional ON/OFF enables/disables constant frequency offset

`set_edrate`(*freq_offset: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:FOFFset:EDRate
driver.configure.rfSettings.dtx.sing.foffset.set_edrate(freq_offset = 1)
```

Specify the constant frequency offset to be added to the center frequency. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param freq_offset numeric | ON | OFF Range: -250 kHz to 250 kHz Additional ON/OFF enables/disables constant frequency offset

Cloning the Group

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

Subgroups

7.1.11.1.2.28 Nmode

class Nmode
Nmode commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.11.1.2.29 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:FOFFset:NMODe:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:FOFFset:NMODe:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:FOFFset:NMODe:LENergy:LE1M
```

class LowEnergy
LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:FOFFset:NMODe:LENergy:LE1M
value: int or bool = driver.configure.rfSettings.dtx.sing.foffset.nmode.
↪lowEnergy.get_le_1_m()
```

Specify the constant frequency offset to be added to the center frequency. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_offset: numeric | ON | OFF Range: -250 kHz to 250 kHz Additional ON/OFF enables/disables constant frequency offset

get_le_2_m() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:FOFFset:NMODe:LENergy:LE2M
value: int or bool = driver.configure.rfSettings.dtx.sing.foffset.nmode.
↪lowEnergy.get_le_2_m()
```

Specify the constant frequency offset to be added to the center frequency. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_offset: numeric | ON | OFF Range: -250 kHz to 250 kHz Additional ON/OFF enables/disables constant frequency offset

get_lrange() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:FOFFset:NMODe:LENergy:LRANge
value: int or bool = driver.configure.rfSettings.dtx.sing.foffset.nmode.
↪lowEnergy.get_lrange()
```

Specify the constant frequency offset to be added to the center frequency. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_offset: numeric | ON | OFF Range: -250 kHz to 250 kHz Additional ON/OFF enables/disables constant frequency offset

set_le_1_m(*freq_offset: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:FOFFset:NMODe:LENergy:LE1M
driver.configure.rfSettings.dtx.sing.foffset.nmode.lowEnergy.set_le_1_m(freq_
↪offset = 1)
```

Specify the constant frequency offset to be added to the center frequency. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param freq_offset numeric | ON | OFF Range: -250 kHz to 250 kHz Additional ON/OFF enables/disables constant frequency offset

set_le_2_m(*freq_offset: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:FOFFset:NMODe:LENergy:LE2M
driver.configure.rfSettings.dtx.sing.foffset.nmode.lowEnergy.set_le_2_m(freq_
↪offset = 1)
```

Specify the constant frequency offset to be added to the center frequency. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param freq_offset numeric | ON | OFF Range: -250 kHz to 250 kHz Additional ON/OFF enables/disables constant frequency offset

`set_lrange`(*freq_offset: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:FOFFset:NMODe:LENergy:LRANge
driver.configure.rfSettings.dtx.sing.foffset.nmode.lowEnergy.set_lrange(freq_
↪offset = 1)
```

Specify the constant frequency offset to be added to the center frequency. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param freq_offset numeric | ON | OFF Range: -250 kHz to 250 kHz Additional ON/OFF enables/disables constant frequency offset

7.1.11.1.2.30 Tmode

class Tmode

Tmode commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.11.1.2.31 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:FOFFset:TMODe:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:FOFFset:TMODe:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:FOFFset:TMODe:LENergy:LE1M
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:FOFFset:TMODe:LENergy:LE1M
value: int or bool = driver.configure.rfSettings.dtx.sing.foffset.tmode.
↪lowEnergy.get_le_1_m()
```

Specify the constant frequency offset to be added to the center frequency. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_offset: numeric | ON | OFF Range: -250 kHz to 250 kHz Additional ON/OFF enables/disables constant frequency offset

get_le_2_m() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:FOFFset:TMODe:LENergy:LE2M
value: int or bool = driver.configure.rfSettings.dtx.sing.foffset.tmode.
↪lowEnergy.get_le_2_m()
```

Specify the constant frequency offset to be added to the center frequency. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_offset: numeric | ON | OFF Range: -250 kHz to 250 kHz Additional ON/OFF enables/disables constant frequency offset

get_lrange() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:FOFFset:TMODe:LENergy:LRANge
value: int or bool = driver.configure.rfSettings.dtx.sing.foffset.tmode.
↪lowEnergy.get_lrange()
```

Specify the constant frequency offset to be added to the center frequency. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_offset: numeric | ON | OFF Range: -250 kHz to 250 kHz Additional ON/OFF enables/disables constant frequency offset

set_le_1_m(*freq_offset: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:FOFFset:TMODe:LENergy:LE1M
driver.configure.rfSettings.dtx.sing.foffset.tmode.lowEnergy.set_le_1_m(freq_
↪offset = 1)
```

Specify the constant frequency offset to be added to the center frequency. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param freq_offset numeric | ON | OFF Range: -250 kHz to 250 kHz Additional ON/OFF enables/disables constant frequency offset

set_le_2_m(*freq_offset: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:FOFFset:TMODe:LENergy:LE2M
driver.configure.rfSettings.dtx.sing.foffset.tmode.lowEnergy.set_le_2_m(freq_
↪offset = 1)
```

Specify the constant frequency offset to be added to the center frequency. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param freq_offset numeric | ON | OFF Range: -250 kHz to 250 kHz Additional ON/OFF enables/disables constant frequency offset

`set_lrange`(*freq_offset: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:FOFFset:TMODe:LENergy:LRANge
driver.configure.rfSettings.dtx.sing.foffset.tmode.lowEnergy.set_lrange(freq_
↪offset = 1)
```

Specify the constant frequency offset to be added to the center frequency. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param freq_offset numeric | ON | OFF Range: -250 kHz to 250 kHz Additional ON/OFF enables/disables constant frequency offset

7.1.11.1.2.32 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:FOFFset:LENergy:LE1M
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:FOFFset:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:SING:FOFFset:LENergy:LE2M
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:FOFFset:LENergy[:LE1M]
value: int or bool = driver.configure.rfSettings.dtx.sing.foffset.lowEnergy.get_
↪le_1_m()
```

Specify the constant frequency offset to be added to the center frequency. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_offset: numeric | ON | OFF Range: -250 kHz to 250 kHz Additional ON/OFF enables/disables constant frequency offset

get_le_2_m() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:FOFFset:LENergy:LE2M
value: int or bool = driver.configure.rfSettings.dtx.sing.foffset.lowEnergy.get_
↪le_2_m()
```

Specify the constant frequency offset to be added to the center frequency. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_offset: numeric | ON | OFF Range: -250 kHz to 250 kHz Additional ON/OFF enables/disables constant frequency offset

`get_lrange`() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:FOFFset:LENergy:LRANge
value: int or bool = driver.configure.rfSettings.dtx.sing.foffset.lowEnergy.get_
↪lrange()
```

Specify the constant frequency offset to be added to the center frequency. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return freq_offset: numeric | ON | OFF Range: -250 kHz to 250 kHz Additional ON/OFF enables/disables constant frequency offset

`set_le_1_m`(*freq_offset: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:FOFFset:LENergy[:LE1M]
driver.configure.rfSettings.dtx.sing.foffset.lowEnergy.set_le_1_m(freq_offset =␣
↪1)
```

Specify the constant frequency offset to be added to the center frequency. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param freq_offset numeric | ON | OFF Range: -250 kHz to 250 kHz Additional ON/OFF enables/disables constant frequency offset

`set_le_2_m`(*freq_offset: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:FOFFset:LENergy:LE2M
driver.configure.rfSettings.dtx.sing.foffset.lowEnergy.set_le_2_m(freq_offset =␣
↪1)
```

Specify the constant frequency offset to be added to the center frequency. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param freq_offset numeric | ON | OFF Range: -250 kHz to 250 kHz Additional ON/OFF enables/disables constant frequency offset

`set_lrange`(*freq_offset: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:SING:FOFFset:LENergy:LRANge
driver.configure.rfSettings.dtx.sing.foffset.lowEnergy.set_lrange(freq_offset =␣
↪1)
```

Specify the constant frequency offset to be added to the center frequency. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param freq_offset numeric | ON | OFF Range: -250 kHz to 250 kHz Additional ON/OFF enables/disables constant frequency offset

7.1.11.1.3 ModFrequency

class ModFrequency

ModFrequency commands group definition. 9 total commands, 3 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.11.1.3.1 Nmode

class Nmode

Nmode commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.11.1.3.2 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:MODFrequency:NMODe:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:MODFrequency:NMODe:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:MODFrequency:NMODe:LENergy:LE1M
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → RsCmwBluetoothSig.enums.DriftRate

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MODFrequency:NMODe:LENergy:LE1M
value: enums.DriftRate = driver.configure.rfSettings.dtx.modFrequency.nmode.
↪lowEnergy.get_le_1_m()
```

Specifies the drift rate for LE dirty transmitter. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return drift_rate: HDRF | LDRF HDRF: 1250 Hz (high drift rate) LDRF: 625 Hz (low drift rate)

get_le_2_m() → RsCmwBluetoothSig.enums.DriftRate

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MODFrequency:NMODe:LENergy:LE2M
value: enums.DriftRate = driver.configure.rfSettings.dtx.modFrequency.nmode.
↪lowEnergy.get_le_2_m()
```

Specifies the drift rate for LE dirty transmitter. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return drift_rate: HDRF | LDRF HDRF: 1250 Hz (high drift rate) LDRF: 625 Hz (low drift rate)

get_lrange() → RsCmwBluetoothSig.enums.DriftRate

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MODFrequency:NMODe:LENergy:LRANge
value: enums.DriftRate = driver.configure.rfSettings.dtx.modFrequency.nmode.
↪lowEnergy.get_lrange()
```

Specifies the drift rate for LE dirty transmitter. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return drift_rate: HDRF | LDRF HDRF: 1250 Hz (high drift rate) LDRF: 625 Hz (low drift rate)

set_le_1_m(*drift_rate: RsCmwBluetoothSig.enums.DriftRate*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MODFrequency:NMODe:LENergy:LE1M
driver.configure.rfSettings.dtx.modFrequency.nmode.lowEnergy.set_le_1_m(drift_
↪rate = enums.DriftRate.HDRF)
```

Specifies the drift rate for LE dirty transmitter. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param drift_rate HDRF | LDRF HDRF: 1250 Hz (high drift rate) LDRF: 625 Hz (low drift rate)

set_le_2_m(*drift_rate: RsCmwBluetoothSig.enums.DriftRate*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MODFrequency:NMODe:LENergy:LE2M
driver.configure.rfSettings.dtx.modFrequency.nmode.lowEnergy.set_le_2_m(drift_
↪rate = enums.DriftRate.HDRF)
```

Specifies the drift rate for LE dirty transmitter. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param drift_rate HDRF | LDRF HDRF: 1250 Hz (high drift rate) LDRF: 625 Hz (low drift rate)

set_lrange(*drift_rate: RsCmwBluetoothSig.enums.DriftRate*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MODFrequency:NMODe:LENergy:LRANge
driver.configure.rfSettings.dtx.modFrequency.nmode.lowEnergy.set_lrange(drift_
↪rate = enums.DriftRate.HDRF)
```

Specifies the drift rate for LE dirty transmitter. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param drift_rate HDRF | LDRF HDRF: 1250 Hz (high drift rate) LDRF: 625 Hz (low drift rate)

7.1.11.1.3.3 Tmode

class Tmode

Tmode commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.11.1.3.4 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:MODFrequency:TMODe:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:MODFrequency:TMODe:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:MODFrequency:TMODe:LENergy:LE1M
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → RsCmwBluetoothSig.enums.DriftRate

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MODFrequency:TMODe:LENergy:LE1M
value: enums.DriftRate = driver.configure.rfSettings.dtx.modFrequency.tmode.
↪lowEnergy.get_le_1_m()
```

Specifies the drift rate for LE dirty transmitter. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return drift_rate: HDRF | LDRF HDRF: 1250 Hz (high drift rate) LDRF: 625 Hz (low drift rate)

get_le_2_m() → RsCmwBluetoothSig.enums.DriftRate

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MODFrequency:TMODe:LENergy:LE2M
value: enums.DriftRate = driver.configure.rfSettings.dtx.modFrequency.tmode.
↪lowEnergy.get_le_2_m()
```

Specifies the drift rate for LE dirty transmitter. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return drift_rate: HDRF | LDRF HDRF: 1250 Hz (high drift rate) LDRF: 625 Hz (low drift rate)

get_lrange() → RsCmwBluetoothSig.enums.DriftRate

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MODFrequency:TMODe:LENergy:LRANge
value: enums.DriftRate = driver.configure.rfSettings.dtx.modFrequency.tmode.
↪lowEnergy.get_lrange()
```

Specifies the drift rate for LE dirty transmitter. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return drift_rate: HDRF | LDRF HDRF: 1250 Hz (high drift rate) LDRF: 625 Hz (low drift rate)

set_le_1_m(*drift_rate: RsCmwBluetoothSig.enums.DriftRate*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MODFrequency:TMODe:LENergy:LE1M
driver.configure.rfSettings.dtx.modFrequency.tmode.lowEnergy.set_le_1_m(drift_
↪rate = enums.DriftRate.HDRF)
```

Specifies the drift rate for LE dirty transmitter. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param drift_rate HDRF | LDRF HDRF: 1250 Hz (high drift rate) LDRF: 625 Hz (low drift rate)

set_le_2_m(*drift_rate: RsCmwBluetoothSig.enums.DriftRate*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MODFrequency:TMODe:LENergy:LE2M
driver.configure.rfSettings.dtx.modFrequency.tmode.lowEnergy.set_le_2_m(drift_
↪rate = enums.DriftRate.HDRF)
```

Specifies the drift rate for LE dirty transmitter. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param drift_rate HDRF | LDRF HDRF: 1250 Hz (high drift rate) LDRF: 625 Hz (low drift rate)

set_lrange(*drift_rate: RsCmwBluetoothSig.enums.DriftRate*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MODFrequency:TMODe:LENergy:LRANge
driver.configure.rfSettings.dtx.modFrequency.tmode.lowEnergy.set_lrange(drift_
↪rate = enums.DriftRate.HDRF)
```

Specifies the drift rate for LE dirty transmitter. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.

- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param drift_rate HDRF | LDRF HDRF: 1250 Hz (high drift rate) LDRF: 625 Hz (low drift rate)

7.1.11.1.3.5 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:MODFrequency:LENergy:LE1M
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:MODFrequency:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:MODFrequency:LENergy:LE2M
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → RsCmwBluetoothSig.enums.DriftRate

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MODFrequency:LENergy[:LE1M]
value: enums.DriftRate = driver.configure.rfSettings.dtx.modFrequency.lowEnergy.
↪get_le_1_m()
```

Specifies the drift rate for LE dirty transmitter. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return drift_rate: HDRF | LDRF HDRF: 1250 Hz (high drift rate) LDRF: 625 Hz (low drift rate)

get_le_2_m() → RsCmwBluetoothSig.enums.DriftRate

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MODFrequency:LENergy:LE2M
value: enums.DriftRate = driver.configure.rfSettings.dtx.modFrequency.lowEnergy.
↪get_le_2_m()
```

Specifies the drift rate for LE dirty transmitter. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return drift_rate: HDRF | LDRF HDRF: 1250 Hz (high drift rate) LDRF: 625 Hz (low drift rate)

get_lrange() → RsCmwBluetoothSig.enums.DriftRate

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MODFrequency:LENergy:LRANge
value: enums.DriftRate = driver.configure.rfSettings.dtx.modFrequency.lowEnergy.
↪get_lrange()
```

Specifies the drift rate for LE dirty transmitter. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return drift_rate: HDRF | LDRF HDRF: 1250 Hz (high drift rate) LDRF: 625 Hz (low drift rate)

set_le_1_m(*drift_rate: RsCmwBluetoothSig.enums.DriftRate*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MODFrequency:LENergy[:LE1M]
driver.configure.rfSettings.dtx.modFrequency.lowEnergy.set_le_1_m(drift_rate =␣
↪enums.DriftRate.HDRF)
```

Specifies the drift rate for LE dirty transmitter. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param drift_rate HDRF | LDRF HDRF: 1250 Hz (high drift rate) LDRF: 625 Hz (low drift rate)

set_le_2_m(*drift_rate: RsCmwBluetoothSig.enums.DriftRate*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MODFrequency:LENergy:LE2M
driver.configure.rfSettings.dtx.modFrequency.lowEnergy.set_le_2_m(drift_rate =␣
↪enums.DriftRate.HDRF)
```

Specifies the drift rate for LE dirty transmitter. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.

- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param drift_rate HDRF | LDRF HDRF: 1250 Hz (high drift rate) LDRF: 625 Hz (low drift rate)

set_lrange(*drift_rate: RsCmwBluetoothSig.enums.DriftRate*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MODFrequency:LENergy:LRANge
driver.configure.rfSettings.dtx.modFrequency.lowEnergy.set_lrange(drift_rate =␣
↪enums.DriftRate.HDRF)
```

Specifies the drift rate for LE dirty transmitter. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param drift_rate HDRF | LDRF HDRF: 1250 Hz (high drift rate) LDRF: 625 Hz (low drift rate)

7.1.11.1.4 Mode

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:MODE:EDRate
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:MODE:BRATe
```

class Mode

Mode commands group definition. 11 total commands, 3 Sub-groups, 2 group commands

get_brate() → RsCmwBluetoothSig.enums.DtxMode

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:MODE:BRATe
value: enums.DtxMode = driver.configure.rfSettings.dtx.mode.get_brate()
```

Configure the dirty transmitter. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return dtx_mode: SINGle values | SPEC table SING: single set of dirty transmitter parameters. No periodic change of the frequency offset, modulation index, and symbol timing error occurs. SPEC: settings according to the test specification for Bluetooth wireless technology, see 'Dirty Tx Mode'.

get_edrate() → RsCmwBluetoothSig.enums.DtxMode

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:MODE:EDRate
value: enums.DtxMode = driver.configure.rfSettings.dtx.mode.get_edrate()
```

Configure the dirty transmitter. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return dtx_mode: SINGle values | SPEC table SING: single set of dirty transmitter parameters. No periodic change of the frequency offset, modulation index, and symbol timing error occurs. SPEC: settings according to the test specification for Bluetooth wireless technology, see 'Dirty Tx Mode'.

set_brate(*dtx_mode: RsCmwBluetoothSig.enums.DtxMode*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:MODE:BRATe
driver.configure.rfSettings.dtx.mode.set_brate(dtx_mode = enums.DtxMode.SINGle)
```

Configure the dirty transmitter. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param dtx_mode SINGle values | SPEC table SING: single set of dirty transmitter parameters. No periodic change of the frequency offset, modulation index, and symbol timing error occurs. SPEC: settings according to the test specification for Bluetooth wireless technology, see 'Dirty Tx Mode'.

set_edrate(*dtx_mode: RsCmwBluetoothSig.enums.DtxMode*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:MODE:EDRate
driver.configure.rfSettings.dtx.mode.set_edrate(dtx_mode = enums.DtxMode.SINGle)
```

Configure the dirty transmitter. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param dtx_mode SINGle values | SPEC table SING: single set of dirty transmitter parameters. No periodic change of the frequency offset, modulation index, and symbol timing error occurs. SPEC: settings according to the test specification for Bluetooth wireless technology, see 'Dirty Tx Mode'.

Cloning the Group

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

Subgroups

7.1.11.1.4.1 Nmode

class Nmode
Nmode commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.11.1.4.2 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:MODE:NMODe:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:MODE:NMODe:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:MODE:NMODe:LENergy:LE1M
```

class LowEnergy
LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → RsCmwBluetoothSig.enums.DtxMode

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MODE:NMODe:LENergy:LE1M
value: enums.DtxMode = driver.configure.rfSettings.dtx.mode.nmode.lowEnergy.get_
↪le_1_m()
```

Configure the dirty transmitter. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return dtx_mode: SINGle values | SPEC table SING: single set of dirty transmitter parameters. No periodic change of the frequency offset, modulation index, and symbol timing error occurs. SPEC: settings according to the test specification for Bluetooth wireless technology, see 'Dirty Tx Mode'.

`get_le_2_m()` → RsCmwBluetoothSig.enums.DtxMode

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MODE:NMODe:LENergy:LE2M
value: enums.DtxMode = driver.configure.rfSettings.dtx.mode.nmode.lowEnergy.get_
↪le_2_m()
```

Configure the dirty transmitter. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return dtx_mode: SINGle values | SPEC table SING: single set of dirty transmitter parameters. No periodic change of the frequency offset, modulation index, and symbol timing error occurs. SPEC: settings according to the test specification for Bluetooth wireless technology, see 'Dirty Tx Mode'.

`get_lrange()` → RsCmwBluetoothSig.enums.DtxMode

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MODE:NMODe:LENergy:LRANge
value: enums.DtxMode = driver.configure.rfSettings.dtx.mode.nmode.lowEnergy.get_
↪lrange()
```

Configure the dirty transmitter. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return dtx_mode: SINGle values | SPEC table SING: single set of dirty transmitter parameters. No periodic change of the frequency offset, modulation index, and symbol timing error occurs. SPEC: settings according to the test specification for Bluetooth wireless technology, see 'Dirty Tx Mode'.

set_le_1_m(*dtx_mode: RsCmwBluetoothSig.enums.DtxMode*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MODE:NMODe:LENergy:LE1M
driver.configure.rfSettings.dtx.mode.nmode.lowEnergy.set_le_1_m(dtx_mode =␣
↪enums.DtxMode.SINGle)
```

Configure the dirty transmitter. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param dtx_mode SINGle values | SPEC table SING: single set of dirty transmitter parameters. No periodic change of the frequency offset, modulation index, and symbol timing error occurs. SPEC: settings according to the test specification for Bluetooth wireless technology, see 'Dirty Tx Mode'.

set_le_2_m(*dtx_mode: RsCmwBluetoothSig.enums.DtxMode*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MODE:NMODe:LENergy:LE2M
driver.configure.rfSettings.dtx.mode.nmode.lowEnergy.set_le_2_m(dtx_mode =␣
↪enums.DtxMode.SINGle)
```

Configure the dirty transmitter. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param dtx_mode SINGle values | SPEC table SING: single set of dirty transmitter parameters. No periodic change of the frequency offset, modulation index, and symbol timing error occurs. SPEC: settings according to the test specification for Bluetooth wireless technology, see 'Dirty Tx Mode'.

set_lrange(*dtx_mode: RsCmwBluetoothSig.enums.DtxMode*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MODE:NMODe:LENergy:LRANge
driver.configure.rfSettings.dtx.mode.nmode.lowEnergy.set_lrange(dtx_mode =␣
↪enums.DtxMode.SINGle)
```

Configure the dirty transmitter. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param dtx_mode SINGle values | SPEC table SING: single set of dirty transmitter parameters. No periodic change of the frequency offset, modulation index, and symbol timing error occurs. SPEC: settings according to the test specification for Bluetooth wireless technology, see 'Dirty Tx Mode'.

7.1.11.1.4.3 Tmode

class Tmode

Tmode commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.11.1.4.4 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:MODE:TMODe:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:MODE:TMODe:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:MODE:TMODe:LENergy:LE1M
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → RsCmwBluetoothSig.enums.DtxMode

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MODE:TMODe:LENergy:LE1M
value: enums.DtxMode = driver.configure.rfSettings.dtx.mode.tmode.lowEnergy.get_
↪le_1_m()
```

Configure the dirty transmitter. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return dtx_mode: SINGle values | SPEC table SING: single set of dirty transmitter parameters. No periodic change of the frequency offset, modulation index, and symbol timing error occurs. SPEC: settings according to the test specification for Bluetooth wireless technology, see 'Dirty Tx Mode'.

get_le_2_m() → RsCmwBluetoothSig.enums.DtxMode

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MODE:TMODe:LENergy:LE2M
value: enums.DtxMode = driver.configure.rfSettings.dtx.mode.tmode.lowEnergy.get_
↪le_2_m()
```

Configure the dirty transmitter. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return dtx_mode: SINGle values | SPEC table SING: single set of dirty transmitter parameters. No periodic change of the frequency offset, modulation index, and symbol timing error occurs. SPEC: settings according to the test specification for Bluetooth wireless technology, see 'Dirty Tx Mode'.

get_lrange() → RsCmwBluetoothSig.enums.DtxMode

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MODE:TMODe:LENergy:LRANge
value: enums.DtxMode = driver.configure.rfSettings.dtx.mode.tmode.lowEnergy.get_
↪lrange()
```

Configure the dirty transmitter. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return dtx_mode: SINGle values | SPEC table SING: single set of dirty transmitter parameters. No periodic change of the frequency offset, modulation index, and symbol timing error occurs. SPEC: settings according to the test specification for Bluetooth wireless technology, see 'Dirty Tx Mode'.

set_le_1_m(*dtx_mode: RsCmwBluetoothSig.enums.DtxMode*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MODE:TMODe:LENergy:LE1M
driver.configure.rfSettings.dtx.mode.tmode.lowEnergy.set_le_1_m(dtx_mode =␣
↪enums.DtxMode.SINGle)
```

Configure the dirty transmitter. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param dtx_mode SINGle values | SPEC table SING: single set of dirty transmitter parameters. No periodic change of the frequency offset, modulation index, and symbol timing error occurs. SPEC: settings according to the test specification for Bluetooth wireless technology, see 'Dirty Tx Mode'.

set_le_2_m(*dtx_mode: RsCmwBluetoothSig.enums.DtxMode*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MODE:TMODe:LENergy:LE2M
driver.configure.rfSettings.dtx.mode.tmode.lowEnergy.set_le_2_m(dtx_mode =␣
↪enums.DtxMode.SINGle)
```

(continues on next page)

(continued from previous page)

Configure the dirty transmitter. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param dtx_mode SINGle values | SPEC table SING: single set of dirty transmitter parameters. No periodic change of the frequency offset, modulation index, and symbol timing error occurs. SPEC: settings according to the test specification for Bluetooth wireless technology, see 'Dirty Tx Mode'.

`set_lrange`(*dtx_mode: RsCmwBluetoothSig.enums.DtxMode*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MODE:TMODe:LENergy:LRANge
driver.configure.rfSettings.dtx.mode.tmode.lowEnergy.set_lrange(dtx_mode =␣
↪enums.DtxMode.SINGle)
```

Configure the dirty transmitter. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param dtx_mode SINGle values | SPEC table SING: single set of dirty transmitter parameters. No periodic change of the frequency offset, modulation index, and symbol timing error occurs. SPEC: settings according to the test specification for Bluetooth wireless technology, see 'Dirty Tx Mode'.

7.1.11.1.4.5 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:MODE:LENergy:LE1M
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:MODE:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:MODE:LENergy:LE2M
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → RsCmwBluetoothSig.enums.DtxMode

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MODE:LENergy[:LE1M]
value: enums.DtxMode = driver.configure.rfSettings.dtx.mode.lowEnergy.get_le_1_
↪m()
```

Configure the dirty transmitter. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return dtx_mode: SINGle values | SPEC table SING: single set of dirty transmitter parameters. No periodic change of the frequency offset, modulation index, and symbol timing error occurs. SPEC: settings according to the test specification for Bluetooth wireless technology, see 'Dirty Tx Mode'.

get_le_2_m() → RsCmwBluetoothSig.enums.DtxMode

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:MODE:LENergy:LE2M
value: enums.DtxMode = driver.configure.rfSettings.dtx.mode.lowEnergy.get_le_2_
↪m()
```

Configure the dirty transmitter. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return dtx_mode: SINGle values | SPEC table SING: single set of dirty transmitter parameters. No periodic change of the frequency offset, modulation index, and symbol timing error occurs. SPEC: settings according to the test specification for Bluetooth wireless technology, see 'Dirty Tx Mode'.

`get_lrange()` → RsCmwBluetoothSig.enums.DtxMode

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MODE:LENergy:LRANge
value: enums.DtxMode = driver.configure.rfSettings.dtx.mode.lowEnergy.get_
↪lrange()
```

Configure the dirty transmitter. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

return dtx_mode: SINGle values | SPEC table SING: single set of dirty transmitter parameters. No periodic change of the frequency offset, modulation index, and symbol timing error occurs. SPEC: settings according to the test specification for Bluetooth wireless technology, see 'Dirty Tx Mode'.

set_le_1_m(*dtx_mode: RsCmwBluetoothSig.enums.DtxMode*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MODE:LENergy[:LE1M]
driver.configure.rfSettings.dtx.mode.lowEnergy.set_le_1_m(dtx_mode = enums.
↪DtxMode.SINGle)
```

Configure the dirty transmitter. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param dtx_mode SINGle values | SPEC table SING: single set of dirty transmitter parameters. No periodic change of the frequency offset, modulation index, and symbol timing error occurs. SPEC: settings according to the test specification for Bluetooth wireless technology, see 'Dirty Tx Mode'.

set_le_2_m(*dtx_mode: RsCmwBluetoothSig.enums.DtxMode*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:MODE:LENergy:LE2M
driver.configure.rfSettings.dtx.mode.lowEnergy.set_le_2_m(dtx_mode = enums.
↪DtxMode.SINGle)
```

Configure the dirty transmitter. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param dtx_mode SINGle values | SPEC table SING: single set of dirty transmitter parameters. No periodic change of the frequency offset, modulation index, and symbol timing error occurs. SPEC: settings according to the test specification for Bluetooth wireless technology, see 'Dirty Tx Mode'.

set_lrange(*dtx_mode: RsCmwBluetoothSig.enums.DtxMode*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MODE:LENergy:LRANge
driver.configure.rfSettings.dtx.mode.lowEnergy.set_lrange(dtx_mode = enums.
↪DtxMode.SINGle)
```

Configure the dirty transmitter. INTRO_CMD_HELP: Refer also to the following commands:

- RF tests:

Commands for test mode classic (..:BRATe..) , (..:EDRate..) and for LE direct test mode (..:LE1M..) , (..:LE2M..) , (.. :LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE connection tests (normal mode) :

Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (.. :NMODe:LENergy:LRANge..) are available.

INTRO_CMD_HELP: Refer also to the following commands:

- LE test mode:

Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (.. :TMODe:LENergy:LRANge..) are available.

param dtx_mode SINGle values | SPEC table SING: single set of dirty transmitter parameters. No periodic change of the frequency offset, modulation index, and symbol timing error occurs. SPEC: settings according to the test specification for Bluetooth wireless technology, see 'Dirty Tx Mode'.

7.1.11.1.5 Mindex

class Mindex

Mindex commands group definition. 6 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.11.1.5.1 Mode

class Mode
Mode commands group definition. 6 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.11.1.5.2 Tmode

class Tmode
Tmode commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.11.1.5.3 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:MINDex:MODE:TMODe:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:MINDex:MODE:TMODe:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:MINDex:MODE:TMODe:LENergy:LE1M
```

class LowEnergy
LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → RsCmwBluetoothSig.enums.ModIndexType

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MINDex:MODE:TMODe:LENergy:LE1M
value: enums.ModIndexType = driver.configure.rfSettings.dtx.mindex.mode.tmode.
↪lowEnergy.get_le_1_m()
```

No command help available

return mod_index_type: No help available

get_le_2_m() → RsCmwBluetoothSig.enums.ModIndexType

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MINDex:MODE:TMODe:LENergy:LE2M
value: enums.ModIndexType = driver.configure.rfSettings.dtx.mindex.mode.tmode.
↪lowEnergy.get_le_2_m()
```

No command help available

return mod_index_type: No help available

get_lrange() → RsCmwBluetoothSig.enums.ModIndexType

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MINDex:MODE:TMODe:LENergy:LRANge
value: enums.ModIndexType = driver.configure.rfSettings.dtx.mindex.mode.tmode.
↪lowEnergy.get_lrange()
```

No command help available

return mod_index_type: No help available

set_le_1_m(*mod_index_type: RsCmwBluetoothSig.enums.ModIndexType*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MINDex:MODE:TMODe:LENergy:LE1M
driver.configure.rfSettings.dtx.mindex.mode.tmode.lowEnergy.set_le_1_m(mod_
↪index_type = enums.ModIndexType.STAB)
```

No command help available

param mod_index_type No help available

set_le_2_m(*mod_index_type: RsCmwBluetoothSig.enums.ModIndexType*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MINDex:MODE:TMODe:LENergy:LE2M
driver.configure.rfSettings.dtx.mindex.mode.tmode.lowEnergy.set_le_2_m(mod_
↪index_type = enums.ModIndexType.STAB)
```

No command help available

param mod_index_type No help available

set_lrange(*mod_index_type: RsCmwBluetoothSig.enums.ModIndexType*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MINDex:MODE:TMODe:LENergy:LRANge
driver.configure.rfSettings.dtx.mindex.mode.tmode.lowEnergy.set_lrange(mod_
↪index_type = enums.ModIndexType.STAB)
```

No command help available

param mod_index_type No help available

7.1.11.1.5.4 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:MINDex:MODE:LENergy:LE1M
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:MINDex:MODE:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:MINDex:MODE:LENergy:LE2M
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → RsCmwBluetoothSig.enums.ModIndexType

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MINDex:MODE:LENergy[:LE1M]
value: enums.ModIndexType = driver.configure.rfSettings.dtx.mindex.mode.
↪lowEnergy.get_le_1_m()
```

No command help available

return mod_index_type: No help available

get_le_2_m() → RsCmwBluetoothSig.enums.ModIndexType

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MINDex:MODE:LENergy:LE2M
value: enums.ModIndexType = driver.configure.rfSettings.dtx.mindex.mode.
↪lowEnergy.get_le_2_m()
```

No command help available

return mod_index_type: No help available

get_lrange() → RsCmwBluetoothSig.enums.ModIndexType

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MINDex:MODE:LENergy:LRANge
value: enums.ModIndexType = driver.configure.rfSettings.dtx.mindex.mode.
↪lowEnergy.get_lrange()
```

No command help available

return mod_index_type: No help available

set_le_1_m(*mod_index_type: RsCmwBluetoothSig.enums.ModIndexType*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MINDex:MODE:LENergy[:LE1M]
driver.configure.rfSettings.dtx.mindex.mode.lowEnergy.set_le_1_m(mod_index_type␣
↪= enums.ModIndexType.STAB)
```

No command help available

param mod_index_type No help available

set_le_2_m(*mod_index_type: RsCmwBluetoothSig.enums.ModIndexType*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MINDex:MODE:LENergy:LE2M
driver.configure.rfSettings.dtx.mindex.mode.lowEnergy.set_le_2_m(mod_index_type␣
↪= enums.ModIndexType.STAB)
```

No command help available

param mod_index_type No help available

set_lrange(*mod_index_type: RsCmwBluetoothSig.enums.ModIndexType*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:DTX:MINDex:MODE:LENergy:LRANge
driver.configure.rfSettings.dtx.mindex.mode.lowEnergy.set_lrange(mod_index_type␣
↪= enums.ModIndexType.STAB)
```

No command help available

param mod_index_type No help available

7.1.11.2 Nmode

class Nmode

Nmode commands group definition. 2 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.11.2.1 Hmode

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:NMODe:HMODe:LENergy
```

class Hmode

Hmode commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_low_energy() → RsCmwBluetoothSig.enums.LeHoppingMode

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:NMODe:HMODe:LENergy
value: enums.LeHoppingMode = driver.configure.rfSettings.nmode.hmode.get_low_
↪energy()
```

Specifies hopping for LE connection tests. Channels used in hopping mode 'CH2' depend on the specified measured channel. First hopping channel is identical with measured channel (method RsCmwBluetoothSig.Configure.RfSettings.Nmode.Mchannel. lowEnergy) . The second hopping channel is offset from the first one.

return hopping_mode: CH2 | ALL 'CH2': 2 data channels offset by 10 or 9 channels are used for hopping 'ALL': 37 data channels are used for hopping

set_low_energy(*hopping_mode: RsCmwBluetoothSig.enums.LeHoppingMode*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:NMODe:HMODe:LENergy
driver.configure.rfSettings.nmode.hmode.set_low_energy(hopping_mode = enums.
↪LeHoppingMode.ALL)
```

Specifies hopping for LE connection tests. Channels used in hopping mode 'CH2' depend on the specified measured channel. First hopping channel is identical with measured channel (method RsCmwBluetoothSig.Configure.RfSettings.Nmode.Mchannel. lowEnergy) . The second hopping channel is offset from the first one.

param hopping_mode CH2 | ALL 'CH2': 2 data channels offset by 10 or 9 channels are used for hopping 'ALL': 37 data channels are used for hopping

7.1.11.2.2 Mchannel

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:NMODe:MCHannel:LENergy
```

class Mchannel

Mchannel commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_low_energy() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:NMODe:MCHannel:LENergy
value: int = driver.configure.rfSettings.nmode.mchannel.get_low_energy()
```

Set the single measured channel for Rx measurements with LE connection tests. Channels for hopping mode 'CH2' depend also on this setting. See also method RsCmwBluetoothSig.Configure.RfSettings.Nmode.Hmode.lowEnergy.

return measured_channel: numeric Channel number Range: 1 to 11, 13 to 38

set_low_energy(*measured_channel: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:NMODe:MCHannel:LENergy
driver.configure.rfSettings.nmode.mchannel.set_low_energy(measured_channel = 1)
```

Set the single measured channel for Rx measurements with LE connection tests. Channels for hopping mode 'CH2' depend also on this setting. See also method RsCmwBluetoothSig.Configure.RfSettings.Nmode.Hmode.lowEnergy.

param measured_channel numeric Channel number Range: 1 to 11, 13 to 38

7.1.11.3 Channel

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:CHANnel:TMODe
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:CHANnel:DTMode
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:CHANnel:LOOPback
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:CHANnel:TXTest
```

class Channel

Channel commands group definition. 4 total commands, 0 Sub-groups, 4 group commands

class LoopbackStruct

Structure for reading output parameters. Fields:

- Rx_Chan: int: numeric Range: 0 Ch to 78 Ch
- Tx_Chan: int: numeric Range: 0 Ch to 78 Ch

get_dt_mode() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:CHANnel:DTMode
value: int = driver.configure.rfSettings.channel.get_dt_mode()
```

Configures the channel number for direct test mode.

return rx_tx_chan: numeric Range: 0 Ch to 39 Ch

get_loopback() → LoopbackStruct

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:CHANnel:LOOPback
value: LoopbackStruct = driver.configure.rfSettings.channel.get_loopback()
```

Defines the channels used by the loopback test.

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

get_tmode() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:CHANnel:TMODe
value: int = driver.configure.rfSettings.channel.get_tmode()
```

Sets the RF channel for LE test mode. This mode supports both data and advertising channels.

return rx_tx_chan: numeric Channel number Range: 0 Ch to 39 Ch

get_tx_test() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:CHANnel:TXTest
value: int = driver.configure.rfSettings.channel.get_tx_test()
```

Defines the channels used by the TX test.

return rx_tx_chan: numeric Range: 0 Ch to 78 Ch

set_dt_mode(*rx_tx_chan: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:CHANnel:DTMode
driver.configure.rfSettings.channel.set_dt_mode(rx_tx_chan = 1)
```

Configures the channel number for direct test mode.

param rx_tx_chan numeric Range: 0 Ch to 39 Ch

set_loopback(*value: RsCmwBluetoothSig.Implementations.Configure_.RfSettings_.Channel.Channel.LoopbackStruct*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:CHANnel:LOOPback
driver.configure.rfSettings.channel.set_loopback(value = LoopbackStruct())
```

Defines the channels used by the loopback test.

param value see the help for LoopbackStruct structure arguments.

set_tmode(*rx_tx_chan: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:CHANnel:TMODe
driver.configure.rfSettings.channel.set_tmode(rx_tx_chan = 1)
```

Sets the RF channel for LE test mode. This mode supports both data and advertising channels.

param rx_tx_chan numeric Channel number Range: 0 Ch to 39 Ch

set_tx_test(*rx_tx_chan: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:CHANnel:TXTest
driver.configure.rfSettings.channel.set_tx_test(rx_tx_chan = 1)
```

Defines the channels used by the TX test.

param rx_tx_chan numeric Range: 0 Ch to 78 Ch

7.1.11.4 Frequency

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:FREQuency:TMODe
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:FREQuency:DTMode
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:FREQuency:TXTest
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:FREQuency:LOOPback
```

class Frequency

Frequency commands group definition. 4 total commands, 0 Sub-groups, 4 group commands

class LoopbackStruct

Structure for reading output parameters. Fields:

- Rx_Freq: float: float Range: 2402 MHz to 2480 MHz , Unit: Hz
- Tx_Freq: float: float Range: 2402 MHz to 2480 MHz , Unit: Hz

get_dt_mode() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:FREQuency:DTMode
value: float = driver.configure.rfSettings.frequency.get_dt_mode()
```

Queries the frequency used for direct test mode.

return rx_tx_freq: float Range: 100 MHz to 6 GHz

get_loopback() → LoopbackStruct

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:FREQuency:LOOPback
value: LoopbackStruct = driver.configure.rfSettings.frequency.get_loopback()
```

Queries EUT RX and EUT TX frequencies used by the loopback test.

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

get_tmode() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:FREQuency:TMODe
value: float = driver.configure.rfSettings.frequency.get_tmode()
```

Queries the frequency used for LE test mode.

return rx_tx_freq: float Range: 100 MHz to 6 GHz

get_tx_test() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:FREQuency:TXTest
value: float = driver.configure.rfSettings.frequency.get_tx_test()
```

Queries the frequency used by the TX test.

return rx_tx_freq: float Range: 2402 MHz to 2480 MHz , Unit: Hz

7.1.11.5 AidOverride

class AidOverride

AidOverride commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.11.5.1 Cte

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:AIDoverride:CTE:LENergy
```

class Cte

Cte commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_low_energy() → List[int]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:AIDoverride:CTE:LENergy
value: List[int] = driver.configure.rfSettings.aidOverride.cte.get_low_energy()
```

No command help available

return antenna_id: No help available

set_low_energy(*antenna_id: List[int]*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:AIDoverride:CTE:LENergy
driver.configure.rfSettings.aidOverride.cte.set_low_energy(antenna_id = [1, 2,␣
↪3])
```

No command help available

param antenna_id No help available

7.1.11.6 Goffset

class Goffset
Goffset commands group definition. 2 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.11.6.1 Cte

class Cte
Cte commands group definition. 2 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.11.6.1.1 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:GOFFset:CTE:LENergy:LE1M
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:GOFFset:CTE:LENergy:LE2M
```

class LowEnergy
LowEnergy commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

class Le1MStruct
Structure for reading output parameters. Fields:

- Ant_1_Gain_Offset: float: numeric Range: -20 dB to 6 dB
- Ant_2_Gain_Offset: float: numeric Range: -20 dB to 6 dB
- Ant_3_Gain_Offset: float: numeric Range: -20 dB to 6 dB

class Le2MStruct
Structure for reading output parameters. Fields:

- Ant_1_Gain_Offset: float: numeric Range: -20 dB to 6 dB
- Ant_2_Gain_Offset: float: numeric Range: -20 dB to 6 dB
- Ant_3_Gain_Offset: float: numeric Range: -20 dB to 6 dB

get_le_1_m() → Le1MStruct

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:GOFFset:CTE:LENergy:LE1M
value: Le1MStruct = driver.configure.rfSettings.goffset.cte.lowEnergy.get_le_1_
↪m()
```

Specifies the gain offset for non-reference antennas relative to the gain (or attenuation) of reference antenna for IQ sample dynamic range measurements. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

return structure: for return value, see the help for Le1MStruct structure arguments.

get_le_2_m() → Le2MStruct

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:GOFFset:CTE:LENergy:LE2M
value: Le2MStruct = driver.configure.rfSettings.goffset.cte.lowEnergy.get_le_2_
↪m()
```

Specifies the gain offset for non-reference antennas relative to the gain (or attenuation) of reference antenna for IQ sample dynamic range measurements. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

return structure: for return value, see the help for Le2MStruct structure arguments.

set_le_1_m(*value: RsCmwBluetoothSig.Implementations.Configure_.RfSettings_.Goffset_.Cte_.LowEnergy.LowEnergy.Le1MStruct*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:GOFFset:CTE:LENergy:LE1M
driver.configure.rfSettings.goffset.cte.lowEnergy.set_le_1_m(value =␣
↪Le1MStruct())
```

Specifies the gain offset for non-reference antennas relative to the gain (or attenuation) of reference antenna for IQ sample dynamic range measurements. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

param value see the help for Le1MStruct structure arguments.

set_le_2_m(*value: RsCmwBluetoothSig.Implementations.Configure_.RfSettings_.Goffset_.Cte_.LowEnergy.LowEnergy.Le2MStruct*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:GOFFset:CTE:LENergy:LE2M
driver.configure.rfSettings.goffset.cte.lowEnergy.set_le_2_m(value =␣
↪Le2MStruct())
```

Specifies the gain offset for non-reference antennas relative to the gain (or attenuation) of reference antenna for IQ sample dynamic range measurements. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

param value see the help for Le2MStruct structure arguments.

7.1.11.7 Aoffset

class Aoffset
Aoffset commands group definition. 2 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.11.7.1 InputPy

class InputPy
InputPy commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.11.7.1.1 Cte

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:AOFFset:INPut:CTE:LENergy
```

class Cte
Cte commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class LowEnergyStruct
Structure for reading output parameters. Fields:

- Ant_1_In_Att_Offset: float: No parameter help available
- Ant_2_In_Att_Offset: float: No parameter help available
- Ant_3_In_Att_Offset: float: No parameter help available

get_low_energy() → LowEnergyStruct

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:AOFFset:INPut:CTE:LENergy
value: LowEnergyStruct = driver.configure.rfSettings.aoffset.inputPy.cte.get_
↪low_energy()
```

(continues on next page)

(continued from previous page)

Specifies the offset of external attenuation per EUT antenna relative to the reference antenna. For the reference antenna, the offset is fixed and set to 0 dB. The commands for input and output path are available. An SUA is required.

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

set_low_energy(*value: RsCmwBluetoothSig.Implementations.Configure_.RfSettings_.Aoffset_.InputPy_.Cte.Cte.LowEnergyStruct*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:AOFFset:INPut:CTE:LENergy
driver.configure.rfSettings.aoffset.inputPy.cte.set_low_energy(value =␣
↪LowEnergyStruct())
```

Specifies the offset of external attenuation per EUT antenna relative to the reference antenna. For the reference antenna, the offset is fixed and set to 0 dB. The commands for input and output path are available. An SUA is required.

param value see the help for LowEnergyStruct structure arguments.

7.1.11.7.2 Output

class Output

Output commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.11.7.2.1 Cte

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:AOFFset:OUTPut:CTE:LENergy
```

class Cte

Cte commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class LowEnergyStruct

Structure for reading output parameters. Fields:

- Ant_1_Out_At_Offset: float: numeric Range: -3 dB to 3 dB
- Ant_2_Out_At_Offset: float: numeric Range: -3 dB to 3 dB
- Ant_3_Out_At_Offset: float: numeric Range: -3 dB to 3 dB

get_low_energy() → LowEnergyStruct

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:AOFFset:OUTPut:CTE:LENergy
value: LowEnergyStruct = driver.configure.rfSettings.aoffset.output.cte.get_low_
↪energy()
```

Specifies the offset of external attenuation per EUT antenna relative to the reference antenna. For the reference antenna, the offset is fixed and set to 0 dB. The commands for input and output path are available. An SUA is required.

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

set_low_energy(*value: RsCmwBluetoothSig.Implementations.Configure_.RfSettings_.Aoffset_.Output_.Cte.Cte.LowEnergyStruct*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RFSettings:AOFFset:OUTPut:CTE:LENergy
driver.configure.rfSettings.aoffset.output.cte.set_low_energy(value =␣
↪LowEnergyStruct())
```

Specifies the offset of external attenuation per EUT antenna relative to the reference antenna. For the reference antenna, the offset is fixed and set to 0 dB. The commands for input and output path are available. An SUA is required.

param value see the help for LowEnergyStruct structure arguments.

7.1.11.8 Nantenna

class Nantenna

Nantenna commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.11.8.1 Cte

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:NANTenna:CTE:LENergy
```

class Cte

Cte commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_low_energy() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:NANTenna:CTE:LENergy
value: int = driver.configure.rfSettings.nantenna.cte.get_low_energy()
```

Specifies the number of EUT's antennas. One reference and one non-reference antennas are mandatory.

return no_of_antenna: numeric Range: 2 to 4

set_low_energy(*no_of_antenna: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:NANTenna:CTE:LENergy
driver.configure.rfSettings.nantenna.cte.set_low_energy(no_of_antenna = 1)
```

Specifies the number of EUT's antennas. One reference and one non-reference antennas are mandatory.

param no_of_antenna numeric Range: 2 to 4

7.1.11.9 Eattenuation

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:EATTenuation:OUTPut
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:EATTenuation:INPut
```

class Eattenuation

Eattenuation commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

get_input_py() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:EATTenuation:INPut
value: float = driver.configure.rfSettings.eattenuation.get_input_py()
```

Defines an external attenuation (or gain, if the value is negative) , to be applied to the input connector.

return external_att: numeric Range: -50 dB to 90 dB, Unit: dB

get_output() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:EATTenuation:OUTPut
value: float = driver.configure.rfSettings.eattenuation.get_output()
```

Defines an external attenuation (or gain, if the value is negative) , to be applied to the output connector.

return external_att: numeric Range: -50 dB to 90 dB, Unit: dB

set_input_py(*external_att: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:EATTenuation:INPut
driver.configure.rfSettings.eattenuation.set_input_py(external_att = 1.0)
```

Defines an external attenuation (or gain, if the value is negative) , to be applied to the input connector.

param external_att numeric Range: -50 dB to 90 dB, Unit: dB

set_output(*external_att: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:EATTenuation:OUTPut
driver.configure.rfSettings.eattenuation.set_output(external_att = 1.0)
```

Defines an external attenuation (or gain, if the value is negative) , to be applied to the output connector.

param external_att numeric Range: -50 dB to 90 dB, Unit: dB

7.1.11.10 AfHopping

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:AFHopping:UCHannels
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:AFHopping
```

class AfHopping

AfHopping commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

class ValueStruct

Structure for reading output parameters. Fields:

- Adaptive_Hopping: bool: OFF | ON Disables, enables adaptive hopping.
- Mode: enums.AfHopingMode: EUT | NORM | USER EUT: only the EUT reports bad channels NORM: both, the EUT and instrument report bad channels USER: bad channels specified manually via [CMDLINK: CONFigure:BLUetooth:SIGNi:RFSettings:AFHopping:UCHannels CMDLINK]

get_uchannels() → List[int]

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:AFHopping:UCHannels
value: List[int] = driver.configure.rfSettings.afHopping.get_uchannels()
```

Specifies user-defined channels for adaptive frequency hopping (AFH) . The setting is relevant for mode = USER, see method RsCmwBluetoothSig.Configure.RfSettings.AfHopping.value.

return channel_list: integer 79 comma-separated values, one value per channel: 0: channel is blocked for AFH 1: channel is released for AFH Range: 0 to 1

get_value() → ValueStruct

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:AFHopping
value: ValueStruct = driver.configure.rfSettings.afHopping.get_value()
```

Specifies the parameters of adaptive hopping.

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

set_uchannels(*channel_list: List[int]*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:AFHopping:UCHannels
driver.configure.rfSettings.afHopping.set_uchannels(channel_list = [1, 2, 3])
```

Specifies user-defined channels for adaptive frequency hopping (AFH) . The setting is relevant for mode = USER, see method RsCmwBluetoothSig.Configure.RfSettings.AfHopping.value.

param channel_list integer 79 comma-separated values, one value per channel: 0: channel is blocked for AFH 1: channel is released for AFH Range: 0 to 1

set_value(*value: RsCmwBluetoothSig.Implementations.Configure_.RfSettings_.AfHopping.AfHopping.ValueStruct*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:AFHopping
driver.configure.rfSettings.afHopping.set_value(value = ValueStruct())
```

Specifies the parameters of adaptive hopping.

param value see the help for ValueStruct structure arguments.

7.1.12 RxQuality

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:REPetition
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:TOUT
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SCONdition
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:IQSDump
```

class RxQuality

RxQuality commands group definition. 113 total commands, 10 Sub-groups, 4 group commands

get_iqsdump() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:IQSDump
value: bool = driver.configure.rxQuality.get_iqsdump()
```

Enables or disables dumping of the Rx IQ events received from the EUT via HCI.

return dump_iq_pairs: OFF | ON

get_repetition() → RsCmwBluetoothSig.enums.Repeat

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:REPetition
value: enums.Repeat = driver.configure.rxQuality.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:BLUetooth:SIGN<i>:RXQuality or method RsCmwBluetoothSig.Configure. RxQuality.Packets.bedr to determine the number of transport blocks per single shot.

return repetition: SINGleshot | CONTinuous SINGleshot: single-shot measurement CONTinuous: continuous measurement

get_scondition() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SCONdition
value: int = driver.configure.rxQuality.get_scondition()
```

No command help available

return condition: No help available

get_timeout() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:TOUT
value: float = driver.configure.rxQuality.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 ([ON | OFF] key or [RESTART | STOP] key) . 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_iqsdump(*dump_iq_pairs: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:IQSDump
driver.configure.rxQuality.set_iqsdump(dump_iq_pairs = False)
```

Enables or disables dumping of the Rx IQ events received from the EUT via HCI.

param dump_iq_pairs OFF | ON

set_repetition(*repetition: RsCmwBluetoothSig.enums.Repeat*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:REPetition
driver.configure.rxQuality.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:BLUetooth:SIGN<i>:RXQuality or method RsCmwBluetoothSig.Configure. RxQuality.Packets.bedr to determine the number of transport blocks per single shot.

param repetition SINGleshot | CONTinuous SINGleshot: single-shot measurement CONTinuous: continuous measurement

set_scondition(*condition: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SCONdition
driver.configure.rxQuality.set_scondition(condition = 1)
```

No command help available

param condition No help available

set_timeout(*timeout: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:TOUT
driver.configure.rxQuality.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 ([ON | OFF] key or [RESTART | STOP] key) . 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.rxQuality.clone()
```

Subgroups

7.1.12.1 SmIndex

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SMINdex:LENergy
```

class SmIndex

SmIndex commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_low_energy() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SMINdex:LENergy
value: bool = driver.configure.rxQuality.smIndex.get_low_energy()
```

Selects the standard or stable modulation index.

return mod_index_type: OFF | ON OFF: standard modulation index is used ON: stable modulation index is used

set_low_energy(*mod_index_type: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SMINdex:LENergy
driver.configure.rxQuality.smIndex.set_low_energy(mod_index_type = False)
```

Selects the standard or stable modulation index.

param mod_index_type OFF | ON OFF: standard modulation index is used ON: stable modulation index is used

7.1.12.2 Search

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:TOUT
```

class Search

Search commands group definition. 35 total commands, 4 Sub-groups, 1 group commands

get_timeout() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:TOUT
value: float = driver.configure.rxQuality.search.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 ([ON | OFF] key or [RESTART | STOP] key) . 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_timeout(*timeout: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:TOUT
driver.configure.rxQuality.search.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 ([ON | OFF] key or [RESTART | STOP] key) . 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.rxQuality.search.clone()
```

Subgroups

7.1.12.2.1 Rintegrity

class Rintegrity
Rintegrity commands group definition. 6 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.12.2.1.1 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:RINTegrity:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:RINTegrity:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:RINTegrity:LENergy:LE1M
```

class LowEnergy
LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:RINTegrity:LENergy[:LE1M]
value: bool = driver.configure.rxQuality.search.rintegrity.lowEnergy.get_le_1_
↪m()
```

Sets the ratio of the test packets with correct CRC transmitted by the R&S CMW. Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.

return report_integrity: OFF | ON OFF: 100% of packets generated with correct CRC
ON: 50% of packets generated with correct CRC

get_le_2_m() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:RINTegrity:LENergy:LE2M
value: bool = driver.configure.rxQuality.search.rintegrity.lowEnergy.get_le_2_
↪m()
```

Sets the ratio of the test packets with correct CRC transmitted by the R&S CMW. Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.

return report_integrity: OFF | ON OFF: 100% of packets generated with correct CRC
ON: 50% of packets generated with correct CRC

get_lrange() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:RINTegrity:LENergy:LRANge
value: bool = driver.configure.rxQuality.search.rintegrity.lowEnergy.get_
↪lrange()
```

Sets the ratio of the test packets with correct CRC transmitted by the R&S CMW. Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.

return report_integrity: OFF | ON OFF: 100% of packets generated with correct CRC
ON: 50% of packets generated with correct CRC

set_le_1_m(*report_integrity: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:RINTegrity:LENergy[:LE1M]
driver.configure.rxQuality.search.rintegrity.lowEnergy.set_le_1_m(report_
↪integrity = False)
```

Sets the ratio of the test packets with correct CRC transmitted by the R&S CMW. Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.

param report_integrity OFF | ON OFF: 100% of packets generated with correct CRC
ON: 50% of packets generated with correct CRC

set_le_2_m(*report_integrity: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:RINTegrity:LENergy:LE2M
driver.configure.rxQuality.search.rintegrity.lowEnergy.set_le_2_m(report_
↪integrity = False)
```

Sets the ratio of the test packets with correct CRC transmitted by the R&S CMW. Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.

param report_integrity OFF | ON OFF: 100% of packets generated with correct CRC
ON: 50% of packets generated with correct CRC

set_lrange(*report_integrity: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:RINTegrity:LENergy:LRANge
driver.configure.rxQuality.search.rintegrity.lowEnergy.set_lrange(report_
↪integrity = False)
```

Sets the ratio of the test packets with correct CRC transmitted by the R&S CMW. Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.

param report_integrity OFF | ON OFF: 100% of packets generated with correct CRC
ON: 50% of packets generated with correct CRC

7.1.12.2.1.2 Tmode

class Tmode

Tmode commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.12.2.1.3 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:RINTegrity:TMODe:LENergy:LE1M
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:RINTegrity:TMODe:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:RINTegrity:TMODe:LENergy:LRANge
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:RINTegrity:TMODe:LENergy:LE1M
value: bool = driver.configure.rxQuality.search.rintegrity.tmode.lowEnergy.get_
↪le_1_m()
```

Sets the ratio of the test packets with correct CRC transmitted by the R&S CMW. Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.

return report_integrity: OFF | ON OFF: 100% of packets generated with correct CRC
ON: 50% of packets generated with correct CRC

get_le_2_m() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:RINTegrity:TMODe:LENergy:LE2M
value: bool = driver.configure.rxQuality.search.rintegrity.tmode.lowEnergy.get_
↪le_2_m()
```

Sets the ratio of the test packets with correct CRC transmitted by the R&S CMW. Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.

return report_integrity: OFF | ON OFF: 100% of packets generated with correct CRC
ON: 50% of packets generated with correct CRC

get_lrange() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:RINTegrity:TMODe:LENergy:LRANge
value: bool = driver.configure.rxQuality.search.rintegrity.tmode.lowEnergy.get_
↪lrange()
```

Sets the ratio of the test packets with correct CRC transmitted by the R&S CMW. Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.

return report_integrity: OFF | ON OFF: 100% of packets generated with correct CRC
ON: 50% of packets generated with correct CRC

set_le_1_m(*report_integrity: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:RINTegrity:TMODe:LENergy:LE1M
driver.configure.rxQuality.search.rintegrity.tmode.lowEnergy.set_le_1_m(report_
↪integrity = False)
```

Sets the ratio of the test packets with correct CRC transmitted by the R&S CMW. Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.

param report_integrity OFF | ON OFF: 100% of packets generated with correct CRC
ON: 50% of packets generated with correct CRC

set_le_2_m(*report_integrity: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:RINTegrity:TMODe:LENergy:LE2M
driver.configure.rxQuality.search.rintegrity.tmode.lowEnergy.set_le_2_m(report_
↪integrity = False)
```

Sets the ratio of the test packets with correct CRC transmitted by the R&S CMW. Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.

param report_integrity OFF | ON OFF: 100% of packets generated with correct CRC
ON: 50% of packets generated with correct CRC

set_lrange(*report_integrity: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:RINTegrity:TMODe:LENergy:LRANge
driver.configure.rxQuality.search.rintegrity.tmode.lowEnergy.set_lrange(report_
↪integrity = False)
```

Sets the ratio of the test packets with correct CRC transmitted by the R&S CMW. Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.

param report_integrity OFF | ON OFF: 100% of packets generated with correct CRC
ON: 50% of packets generated with correct CRC

7.1.12.2.2 Limit

class Limit
Limit commands group definition. 14 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.12.2.2.1 Mper

class Mper
Mper commands group definition. 9 total commands, 3 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.12.2.2.2 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:LIMit:MPER:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:LIMit:MPER:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:LIMit:MPER:LENergy:LE1M
```

class LowEnergy
LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:LIMit:MPER:LENergy[:LE1M]
value: float or bool = driver.configure.rxQuality.search.limit.mper.lowEnergy.
↪get_le_1_m()
```

Specifies the upper PER limit for LE PER search measurements. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.

- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return limit: numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default level)

get_le_2_m() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:LIMit:MPER:LENergy:LE2M
value: float or bool = driver.configure.rxQuality.search.limit.mper.lowEnergy.
↪get_le_2_m()
```

Specifies the upper PER limit for LE PER search measurements. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return limit: numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default level)

get_lrange() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:LIMit:MPER:LENergy:LRANge
value: float or bool = driver.configure.rxQuality.search.limit.mper.lowEnergy.
↪get_lrange()
```

Specifies the upper PER limit for LE PER search measurements. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return limit: numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default level)

set_le_1_m(*limit: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:LIMit:MPER:LENergy[:LE1M]
driver.configure.rxQuality.search.limit.mper.lowEnergy.set_le_1_m(limit = 1.0)
```

Specifies the upper PER limit for LE PER search measurements. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param limit numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default level)

set_le_2_m(*limit: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:LIMit:MPER:LENergy:LE2M
driver.configure.rxQuality.search.limit.mper.lowEnergy.set_le_2_m(limit = 1.0)
```

Specifies the upper PER limit for LE PER search measurements. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param limit numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default level)

set_lrange(*limit: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:LIMit:MPER:LENergy:LRANge
driver.configure.rxQuality.search.limit.mper.lowEnergy.set_lrange(limit = 1.0)
```

Specifies the upper PER limit for LE PER search measurements. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param limit numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default level)

7.1.12.2.2.3 Tmode

class Tmode
Tmode commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.12.2.2.4 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:LIMit:MPER:TMODe:LENergy:LE1M
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:LIMit:MPER:TMODe:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:LIMit:MPER:TMODe:LENergy:LRANge
```

class LowEnergy
LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:LIMit:MPER:TMODe:LENergy:LE1M
value: float or bool = driver.configure.rxQuality.search.limit.mper.tmode.
↪lowEnergy.get_le_1_m()
```

Specifies the upper PER limit for LE PER search measurements. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return limit: numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default level)

get_le_2_m() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:LIMit:MPER:TMODe:LENergy:LE2M
value: float or bool = driver.configure.rxQuality.search.limit.mper.tmode.
↪lowEnergy.get_le_2_m()
```

Specifies the upper PER limit for LE PER search measurements. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return limit: numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default level)

get_lrange() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:LIMit:MPER:TMODe:LENergy:LRANge
value: float or bool = driver.configure.rxQuality.search.limit.mper.tmode.
↪lowEnergy.get_lrange()
```

Specifies the upper PER limit for LE PER search measurements. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return limit: numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default level)

set_le_1_m(*limit: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:LIMit:MPER:TMODe:LENergy:LE1M
driver.configure.rxQuality.search.limit.mper.tmode.lowEnergy.set_le_1_m(limit =␣
↪1.0)
```

Specifies the upper PER limit for LE PER search measurements. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param limit numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default level)

set_le_2_m(*limit: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:LIMit:MPER:TMODe:LENergy:LE2M
driver.configure.rxQuality.search.limit.mper.tmode.lowEnergy.set_le_2_m(limit =␣
↪1.0)
```

Specifies the upper PER limit for LE PER search measurements. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param limit numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default level)

`set_lrange`(*limit: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:LIMit:MPER:TMODe:LENergy:LRANge
driver.configure.rxQuality.search.limit.mper.tmode.lowEnergy.set_lrange(limit =␣
↪1.0)
```

Specifies the upper PER limit for LE PER search measurements. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param limit numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default level)

7.1.12.2.2.5 Nmode

`class Nmode`

Nmode commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.12.2.2.6 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:LIMit:MPER:NMODe:LENergy:LE1M
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:LIMit:MPER:NMODe:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:LIMit:MPER:NMODe:LENergy:LRANge
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:LIMit:MPER:NMODe:LENergy:LE1M
value: float or bool = driver.configure.rxQuality.search.limit.mper.nmode.
↪lowEnergy.get_le_1_m()
```

Specifies the upper PER limit for LE PER search measurements. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return limit: numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default level)

get_le_2_m() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:LIMit:MPER:NMODe:LENergy:LE2M
value: float or bool = driver.configure.rxQuality.search.limit.mper.nmode.
↪lowEnergy.get_le_2_m()
```

Specifies the upper PER limit for LE PER search measurements. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return limit: numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default level)

get_lrange() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:LIMit:MPER:NMODe:LENergy:LRANge
value: float or bool = driver.configure.rxQuality.search.limit.mper.nmode.
↪lowEnergy.get_lrange()
```

Specifies the upper PER limit for LE PER search measurements. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return limit: numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default level)

`set_le_1_m`(*limit: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:LIMit:MPER:NMODe:LENergy:LE1M
driver.configure.rxQuality.search.limit.mper.nmode.lowEnergy.set_le_1_m(limit =␣
↪1.0)
```

Specifies the upper PER limit for LE PER search measurements. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param limit numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default level)

`set_le_2_m`(*limit: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:LIMit:MPER:NMODe:LENergy:LE2M
driver.configure.rxQuality.search.limit.mper.nmode.lowEnergy.set_le_2_m(limit =␣
↪1.0)
```

Specifies the upper PER limit for LE PER search measurements. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param limit numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default level)

set_lrange(*limit: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:LIMit:MPER:NMODe:LENergy:LRANge
driver.configure.rxQuality.search.limit.mper.nmode.lowEnergy.set_lrange(limit =␣
↪1.0)
```

Specifies the upper PER limit for LE PER search measurements. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param limit numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default level)

7.1.12.2.2.7 Mber

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:LIMit:MBER:BRATe
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:LIMit:MBER:EDRate
```

class Mber

Mber commands group definition. 5 total commands, 1 Sub-groups, 2 group commands

get_brate() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:LIMit:MBER:BRATe
value: float or bool = driver.configure.rxQuality.search.limit.mber.get_brate()
```

Specifies the upper BER limit for BR (..:BRATe) and EDR (..:EDRate) BER search measurements.

return limit: numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables | enables the limit using the previous/default value)

get_edrate() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:LIMit:MBER:EDRate
value: float or bool = driver.configure.rxQuality.search.limit.mber.get_edrate()
```

Specifies the upper BER limit for BR (..:BRATe) and EDR (..:EDRate) BER search measurements.

return limit: numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables | enables the limit using the previous/default value)

set_brate(*limit: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:LIMit:MBER:BRATe
driver.configure.rxQuality.search.limit.mber.set_brate(limit = 1.0)
```

Specifies the upper BER limit for BR (..:BRATe) and EDR (..:EDRate) BER search measurements.

param limit numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables | enables the limit using the previous/default value)

set_edrate(*limit: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:LIMit:MBER:EDRate
driver.configure.rxQuality.search.limit.mber.set_edrate(limit = 1.0)
```

Specifies the upper BER limit for BR (..:BRATe) and EDR (..:EDRate) BER search measurements.

param limit numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables | enables the limit using the previous/default value)

Cloning the Group

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

Subgroups

7.1.12.2.2.8 Tmode

class Tmode

Tmode commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.12.2.2.9 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:LIMit:MBER:TMODe:LENergy:LE1M
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:LIMit:MBER:TMODe:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:LIMit:MBER:TMODe:LENergy:LRANge
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:LIMit:MBER:TMODe:LENergy:LE1M
value: float or bool = driver.configure.rxQuality.search.limit.mber.tmode.
↪lowEnergy.get_le_1_m()
```

Specifies the upper BER limit for RX search measurements in LE test mode. Commands for uncoded LE 1M PHY (.. :TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (..:TMODe:LENergy:LRANge..) are available.

return limit: numeric | ON | OFF Range: 0 % to 100 % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default value)

get_le_2_m() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:LIMit:MBER:TMODe:LENergy:LE2M
value: float or bool = driver.configure.rxQuality.search.limit.mber.tmode.
↪lowEnergy.get_le_2_m()
```

Specifies the upper BER limit for RX search measurements in LE test mode. Commands for uncoded LE 1M PHY (.. :TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (..:TMODe:LENergy:LRANge..) are available.

return limit: numeric | ON | OFF Range: 0 % to 100 % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default value)

get_lrange() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:LIMit:MBER:TMODe:LENergy:LRANge
value: float or bool = driver.configure.rxQuality.search.limit.mber.tmode.
↪lowEnergy.get_lrange()
```

Specifies the upper BER limit for RX search measurements in LE test mode. Commands for uncoded LE 1M PHY (.. :TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (..:TMODe:LENergy:LRANge..) are available.

return limit: numeric | ON | OFF Range: 0 % to 100 % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default value)

set_le_1_m(*limit: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:LIMit:MBER:TMODe:LENergy:LE1M
driver.configure.rxQuality.search.limit.mber.tmode.lowEnergy.set_le_1_m(limit =␣
↪1.0)
```

Specifies the upper BER limit for RX search measurements in LE test mode. Commands for uncoded LE 1M PHY (.. :TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (..:TMODe:LENergy:LRANge..) are available.

param limit numeric | ON | OFF Range: 0 % to 100 % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default value)

set_le_2_m(*limit: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:LIMit:MBER:TMODe:LENergy:LE2M
driver.configure.rxQuality.search.limit.mber.tmode.lowEnergy.set_le_2_m(limit =␣
↪1.0)
```

Specifies the upper BER limit for RX search measurements in LE test mode. Commands for uncoded LE 1M PHY (.. :TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (..:TMODe:LENergy:LRANge..) are available.

param limit numeric | ON | OFF Range: 0 % to 100 % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default value)

set_lrange(*limit: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:LIMit:MBER:TMODe:LENergy:LRANge
driver.configure.rxQuality.search.limit.mber.tmode.lowEnergy.set_lrange(limit =␣
↪1.0)
```

Specifies the upper BER limit for RX search measurements in LE test mode. Commands for uncoded LE 1M PHY (.. :TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (..:TMODe:LENergy:LRANge..) are available.

param limit numeric | ON | OFF Range: 0 % to 100 % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default value)

7.1.12.2.3 Packets

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PACKets:BEDR
```

class Packets

Packets commands group definition. 10 total commands, 3 Sub-groups, 1 group commands

get_bedr() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PACKets[:BEDR]
value: int = driver.configure.rxQuality.search.packets.get_bedr()
```

Defines the number of data packets to be measured per measurement cycle (statistics cycle) .

return number_packets: numeric Range: 1 to 400E+3

set_bedr(*number_packets: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PACKets[:BEDR]
driver.configure.rxQuality.search.packets.set_bedr(number_packets = 1)
```

Defines the number of data packets to be measured per measurement cycle (statistics cycle) .

param number_packets numeric Range: 1 to 400E+3

Cloning the Group

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

Subgroups

7.1.12.2.3.1 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PACKets:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PACKets:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PACKets:LENergy:LE1M
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:PACKets:LENergy[:LE1M]
value: int = driver.configure.rxQuality.search.packets.lowEnergy.get_le_1_m()
```

Defines the number of data packets to be measured per measurement cycle (statistics cycle) .

INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return number_packets: numeric Range: 1 to 30E+3

get_le_2_m() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:PACKets:LENergy:LE2M
value: int = driver.configure.rxQuality.search.packets.lowEnergy.get_le_2_m()
```

Defines the number of data packets to be measured per measurement cycle (statistics cycle) .
INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return number_packets: numeric Range: 1 to 30E+3

get_lrange() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:PACKets:LENergy:LRANge
value: int = driver.configure.rxQuality.search.packets.lowEnergy.get_lrange()
```

Defines the number of data packets to be measured per measurement cycle (statistics cycle) .
INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return number_packets: numeric Range: 1 to 30E+3

set_le_1_m(*number_packets: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:PACKets:LENergy[:LE1M]
driver.configure.rxQuality.search.packets.lowEnergy.set_le_1_m(number_packets =␣
↪1)
```

Defines the number of data packets to be measured per measurement cycle (statistics cycle) .
INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param number_packets numeric Range: 1 to 30E+3

set_le_2_m(*number_packets: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:PACKets:LENergy:LE2M
driver.configure.rxQuality.search.packets.lowEnergy.set_le_2_m(number_packets =␣
↪1)
```

Defines the number of data packets to be measured per measurement cycle (statistics cycle) .
INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param number_packets numeric Range: 1 to 30E+3

set_lrange(*number_packets: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:PACKets:LENergy:LRANge
driver.configure.rxQuality.search.packets.lowEnergy.set_lrange(number_packets =␣
↪1)
```

Defines the number of data packets to be measured per measurement cycle (statistics cycle) .
INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param number_packets numeric Range: 1 to 30E+3

7.1.12.2.3.2 Tmode

class Tmode
Tmode commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.12.2.3.3 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PACKets:TMODe:LENergy:LE1M
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PACKets:TMODe:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PACKets:TMODe:LENergy:LRANge
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:PACKets:TMODe:LENergy:LE1M
value: int = driver.configure.rxQuality.search.packets.tmode.lowEnergy.get_le_1_
↪m()
```

Defines the number of data packets to be measured per measurement cycle (statistics cycle) .

INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return number_packets: numeric Range: 1 to 30E+3

get_le_2_m() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:PACKets:TMODe:LENergy:LE2M
value: int = driver.configure.rxQuality.search.packets.tmode.lowEnergy.get_le_2_
↪m()
```

Defines the number of data packets to be measured per measurement cycle (statistics cycle) .

INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return number_packets: numeric Range: 1 to 30E+3

get_lrange() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:PACKets:TMODe:LENergy:LRANge
value: int = driver.configure.rxQuality.search.packets.tmode.lowEnergy.get_
↪lrange()
```

Defines the number of data packets to be measured per measurement cycle (statistics cycle) .
INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return number_packets: numeric Range: 1 to 30E+3

set_le_1_m(*number_packets: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:PACKets:TMODe:LENergy:LE1M
driver.configure.rxQuality.search.packets.tmode.lowEnergy.set_le_1_m(number_
↪packets = 1)
```

Defines the number of data packets to be measured per measurement cycle (statistics cycle) .
INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param number_packets numeric Range: 1 to 30E+3

set_le_2_m(*number_packets: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:PACKets:TMODe:LENergy:LE2M
driver.configure.rxQuality.search.packets.tmode.lowEnergy.set_le_2_m(number_
↪packets = 1)
```

Defines the number of data packets to be measured per measurement cycle (statistics cycle) .
INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param number_packets numeric Range: 1 to 30E+3

set_lrange(*number_packets: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:PACKets:TMODe:LENergy:LRANge
driver.configure.rxQuality.search.packets.tmode.lowEnergy.set_lrange(number_
↪packets = 1)
```

Defines the number of data packets to be measured per measurement cycle (statistics cycle) .

INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param number_packets numeric Range: 1 to 30E+3

7.1.12.2.3.4 Nmode

class Nmode

Nmode commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.12.2.3.5 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PACKets:NMODe:LENergy:LE1M
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PACKets:NMODe:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PACKets:NMODe:LENergy:LRANge
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:PACKets:NMODe:LENergy:LE1M
value: int = driver.configure.rxQuality.search.packets.nmode.lowEnergy.get_le_1_
↪m()
```

Defines the number of data packets to be measured per measurement cycle (statistics cycle) .
INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return number_packets: numeric Range: 1 to 30E+3

get_le_2_m() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:PACKets:NMODe:LENergy:LE2M
value: int = driver.configure.rxQuality.search.packets.nmode.lowEnergy.get_le_2_
↪m()
```

Defines the number of data packets to be measured per measurement cycle (statistics cycle) .
INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return number_packets: numeric Range: 1 to 30E+3

get_lrange() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:PACKets:NMODe:LENergy:LRANge
value: int = driver.configure.rxQuality.search.packets.nmode.lowEnergy.get_
↪lrange()
```

Defines the number of data packets to be measured per measurement cycle (statistics cycle) .
INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return number_packets: numeric Range: 1 to 30E+3

set_le_1_m(*number_packets: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:PACKets:NMODe:LENergy:LE1M
driver.configure.rxQuality.search.packets.nmode.lowEnergy.set_le_1_m(number_
↪packets = 1)
```

Defines the number of data packets to be measured per measurement cycle (statistics cycle) .
INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param number_packets numeric Range: 1 to 30E+3

`set_le_2_m`(*number_packets: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:PACKets:NMODe:LENergy:LE2M
driver.configure.rxQuality.search.packets.nmode.lowEnergy.set_le_2_m(number_
↪packets = 1)
```

Defines the number of data packets to be measured per measurement cycle (statistics cycle) .
INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param number_packets numeric Range: 1 to 30E+3

`set_lrange`(*number_packets: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:PACKets:NMODe:LENergy:LRANge
driver.configure.rxQuality.search.packets.nmode.lowEnergy.set_lrange(number_
↪packets = 1)
```

Defines the number of data packets to be measured per measurement cycle (statistics cycle) .
INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param number_packets numeric Range: 1 to 30E+3

7.1.12.2.4 Step

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:STEP:BREDr
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:STEP:LENergy
```

class Step

Step commands group definition. 4 total commands, 2 Sub-groups, 2 group commands

get_bredr() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:STEP:BREDr
value: float or bool = driver.configure.rxQuality.search.step.get_bredr()
```

Specifies the power step for the BR/EDR search iteration of BER search measurements.

return level_step: numeric Range: 0.01 dBm to 5 dBm, Unit: dB

get_low_energy() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:STEP:LENergy
value: float or bool = driver.configure.rxQuality.search.step.get_low_energy()
```

Specifies the power step for the LE search iteration of PER search measurements. INTRO_CMD_HELP: Refer also to the following commands:

- For LE connection tests (normal mode) , command for LE 1M PHY - uncoded (..:NMODe:LENergy:LE1M..) is available.
- For LE test mode, command ..:SEARch:STEP:TMODe:LENergy.. is available.
- For LE RF tests (direct test mode) , command ..:SEARch:STEP:LENergy.. is available.

return level_step: numeric Range: 0.01 dB to 5 dB, Unit: dB

set_bredr(*level_step: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:STEP:BREDr
driver.configure.rxQuality.search.step.set_bredr(level_step = 1.0)
```

Specifies the power step for the BR/EDR search iteration of BER search measurements.

param level_step numeric Range: 0.01 dBm to 5 dBm, Unit: dB

set_low_energy(*level_step: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:STEP:LENergy
driver.configure.rxQuality.search.step.set_low_energy(level_step = 1.0)
```

Specifies the power step for the LE search iteration of PER search measurements. INTRO_CMD_HELP: Refer also to the following commands:

- For LE connection tests (normal mode) , command for LE 1M PHY - uncoded (..:NMODe:LENergy:LE1M..) is available.
- For LE test mode, command ..:SEARch:STEP:TMODe:LENergy.. is available.
- For LE RF tests (direct test mode) , command ..:SEARch:STEP:LENergy.. is available.

param level_step numeric Range: 0.01 dB to 5 dB, Unit: dB

Cloning the Group

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

Subgroups

7.1.12.2.4.1 Tmode

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:STEP:TMODe:LENergy
```

class Tmode

Tmode commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_low_energy() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:STEP:TMODe:LENergy
value: float or bool = driver.configure.rxQuality.search.step.tmode.get_low_
↪energy()
```

Specifies the power step for the LE search iteration of PER search measurements. INTRO_CMD_HELP: Refer also to the following commands:

- For LE connection tests (normal mode) , command for LE 1M PHY - uncoded (..:NMODe:LENergy:LE1M..) is available.
- For LE test mode, command ..:SEARch:STEP:TMODe:LENergy.. is available.
- For LE RF tests (direct test mode) , command ..:SEARch:STEP:LENergy.. is available.

return level_step: numeric Range: 0.01 dB to 5 dB, Unit: dB

set_low_energy(*level_step: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:STEP:TMODe:LENergy
driver.configure.rxQuality.search.step.tmode.set_low_energy(level_step = 1.0)
```

Specifies the power step for the LE search iteration of PER search measurements. INTRO_CMD_HELP: Refer also to the following commands:

- For LE connection tests (normal mode) , command for LE 1M PHY - uncoded (..:NMODe:LENergy:LE1M..) is available.
- For LE test mode, command ..:SEARch:STEP:TMODe:LENergy.. is available.
- For LE RF tests (direct test mode) , command ..:SEARch:STEP:LENergy.. is available.

param level_step numeric Range: 0.01 dB to 5 dB, Unit: dB

7.1.12.2.4.2 Nmode

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:STEP:NMODe:LENergy
```

class Nmode

Nmode commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_low_energy() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:STEP:NMODe:LENergy
value: float or bool = driver.configure.rxQuality.search.step.nmode.get_low_
↪energy()
```

Specifies the power step for the LE search iteration of PER search measurements. INTRO_CMD_HELP: Refer also to the following commands:

- For LE connection tests (normal mode) , command for LE 1M PHY - uncoded (..:NMODe:LENergy:LE1M..) is available.
- For LE test mode, command ..:SEARch:STEP:TMODe:LENergy.. is available.
- For LE RF tests (direct test mode) , command ..:SEARch:STEP:LENergy.. is available.

return level_step: numeric Range: 0.01 dB to 5 dB, Unit: dB

set_low_energy(*level_step: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:STEP:NMODe:LENergy
driver.configure.rxQuality.search.step.nmode.set_low_energy(level_step = 1.0)
```

Specifies the power step for the LE search iteration of PER search measurements. INTRO_CMD_HELP: Refer also to the following commands:

- For LE connection tests (normal mode) , command for LE 1M PHY - uncoded (..:NMODe:LENergy:LE1M..) is available.
- For LE test mode, command ..:SEARch:STEP:TMODe:LENergy.. is available.
- For LE RF tests (direct test mode) , command ..:SEARch:STEP:LENergy.. is available.

param level_step numeric Range: 0.01 dB to 5 dB, Unit: dB

7.1.12.3 Packets

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:PACKets:BEDR
```

class Packets

Packets commands group definition. 10 total commands, 3 Sub-groups, 1 group commands

get_bedr() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:PACKets[:BEDR]
value: int = driver.configure.rxQuality.packets.get_bedr()
```

Defines the number of data packets to be measured per measurement cycle (statistics cycle) .

return number_packets: numeric Range: 1 to 400E+3

set_bedr(*number_packets: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:PACKets[:BEDR]
driver.configure.rxQuality.packets.set_bedr(number_packets = 1)
```

Defines the number of data packets to be measured per measurement cycle (statistics cycle) .

param number_packets numeric Range: 1 to 400E+3

Cloning the Group

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

Subgroups

7.1.12.3.1 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:PACKets:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:PACKets:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:PACKets:LENergy:LE1M
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:PACKets:LENergy[:LE1M]
value: int = driver.configure.rxQuality.packets.lowEnergy.get_le_1_m()
```

Defines the number of packets to be measured per measurement cycle (statistics cycle) . INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return number_packets: numeric Range: 1 to 30E+3

get_le_2_m() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:PACKets:LENergy:LE2M
value: int = driver.configure.rxQuality.packets.lowEnergy.get_le_2_m()
```

Defines the number of packets to be measured per measurement cycle (statistics cycle) . INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return number_packets: numeric Range: 1 to 30E+3

get_lrange() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:PACKets:LENergy:LRANge
value: int = driver.configure.rxQuality.packets.lowEnergy.get_lrange()
```

Defines the number of packets to be measured per measurement cycle (statistics cycle) . INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return number_packets: numeric Range: 1 to 30E+3

set_le_1_m(*number_packets: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:PACKets:LENergy[:LE1M]
driver.configure.rxQuality.packets.lowEnergy.set_le_1_m(number_packets = 1)
```

Defines the number of packets to be measured per measurement cycle (statistics cycle) . INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.

- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param number_packets numeric Range: 1 to 30E+3

`set_le_2_m`(*number_packets: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:PACKets:LENergy:LE2M
driver.configure.rxQuality.packets.lowEnergy.set_le_2_m(number_packets = 1)
```

Defines the number of packets to be measured per measurement cycle (statistics cycle) . INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param number_packets numeric Range: 1 to 30E+3

`set_lrange`(*number_packets: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:PACKets:LENergy:LRANge
driver.configure.rxQuality.packets.lowEnergy.set_lrange(number_packets = 1)
```

Defines the number of packets to be measured per measurement cycle (statistics cycle) . INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param number_packets numeric Range: 1 to 30E+3

7.1.12.3.2 Tmode

`class Tmode`

Tmode commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.12.3.2.1 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:PACKets:TMODe:LENergy:LE1M
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:PACKets:TMODe:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:PACKets:TMODe:LENergy:LRANge
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:PACKets:TMODe:LENergy:LE1M
value: int = driver.configure.rxQuality.packets.tmode.lowEnergy.get_le_1_m()
```

Defines the number of packets to be measured per measurement cycle (statistics cycle) . INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return number_packets: numeric Range: 1 to 30E+3

get_le_2_m() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:PACKets:TMODe:LENergy:LE2M
value: int = driver.configure.rxQuality.packets.tmode.lowEnergy.get_le_2_m()
```

Defines the number of packets to be measured per measurement cycle (statistics cycle) . INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return number_packets: numeric Range: 1 to 30E+3

get_lrange() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:PACKets:TMODe:LENergy:LRANge
value: int = driver.configure.rxQuality.packets.tmode.lowEnergy.get_lrange()
```

Defines the number of packets to be measured per measurement cycle (statistics cycle) . INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return number_packets: numeric Range: 1 to 30E+3

set_le_1_m(*number_packets: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:PACKets:TMODe:LENergy:LE1M
driver.configure.rxQuality.packets.tmode.lowEnergy.set_le_1_m(number_packets =␣
↪1)
```

Defines the number of packets to be measured per measurement cycle (statistics cycle) . INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param number_packets numeric Range: 1 to 30E+3

set_le_2_m(*number_packets: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:PACKets:TMODe:LENergy:LE2M
driver.configure.rxQuality.packets.tmode.lowEnergy.set_le_2_m(number_packets =␣
↪1)
```

Defines the number of packets to be measured per measurement cycle (statistics cycle) . INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param number_packets numeric Range: 1 to 30E+3

set_lrange(*number_packets: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:PACKets:TMODe:LENergy:LRANge
driver.configure.rxQuality.packets.tmode.lowEnergy.set_lrange(number_packets =␣
↪1)
```

Defines the number of packets to be measured per measurement cycle (statistics cycle) . IN-
TRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param number_packets numeric Range: 1 to 30E+3

7.1.12.3.3 Nmode

class Nmode

Nmode commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.12.3.3.1 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:PACKets:NMODe:LENergy:LE1M
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:PACKets:NMODe:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:PACKets:NMODe:LENergy:LRANge
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:PACKets:NMODe:LENergy:LE1M
value: int = driver.configure.rxQuality.packets.nmode.lowEnergy.get_le_1_m()
```

Defines the number of packets to be measured per measurement cycle (statistics cycle) . IN-
TRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return number_packets: numeric Range: 1 to 30E+3

get_le_2_m() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:PACKets:NMODe:LENergy:LE2M
value: int = driver.configure.rxQuality.packets.nmode.lowEnergy.get_le_2_m()
```

Defines the number of packets to be measured per measurement cycle (statistics cycle) . INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return number_packets: numeric Range: 1 to 30E+3

get_lrange() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:PACKets:NMODe:LENergy:LRANge
value: int = driver.configure.rxQuality.packets.nmode.lowEnergy.get_lrange()
```

Defines the number of packets to be measured per measurement cycle (statistics cycle) . INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return number_packets: numeric Range: 1 to 30E+3

set_le_1_m(*number_packets: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:PACKets:NMODe:LENergy:LE1M
driver.configure.rxQuality.packets.nmode.lowEnergy.set_le_1_m(number_packets =␣
↪1)
```

Defines the number of packets to be measured per measurement cycle (statistics cycle) . INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param number_packets numeric Range: 1 to 30E+3

set_le_2_m(*number_packets: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:PACKets:NMODe:LENergy:LE2M
driver.configure.rxQuality.packets.nmode.lowEnergy.set_le_2_m(number_packets =␣
↪1)
```

Defines the number of packets to be measured per measurement cycle (statistics cycle) . INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param number_packets numeric Range: 1 to 30E+3

set_lrange(*number_packets: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:PACKets:NMODe:LENergy:LRANge
driver.configure.rxQuality.packets.nmode.lowEnergy.set_lrange(number_packets =␣
↪1)
```

Defines the number of packets to be measured per measurement cycle (statistics cycle) . INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param number_packets numeric Range: 1 to 30E+3

7.1.12.4 Rintegrity

class Rintegrity
Rintegrity commands group definition. 6 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.12.4.1 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:RINTegrity:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:RINTegrity:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:RINTegrity:LENergy:LE1M
```

class LowEnergy
LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:RINTegrity:LENergy[:LE1M]
value: bool = driver.configure.rxQuality.rintegrity.lowEnergy.get_le_1_m()
```

Sets the ratio of the test packets with correct CRC transmitted by the R&S CMW. INTRO_CMD_HELP: Refer also to the following commands:

- LE RF tests (direct test mode) : Commands for LE 1M PHY - uncoded (..:LE1M..) , LE 2M PHY - uncoded (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return report_integrity: OFF | ON OFF: 100% of packets generated with correct CRC ON: 50% of packets generated with correct CRC

get_le_2_m() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:RINTegrity:LENergy:LE2M
value: bool = driver.configure.rxQuality.rintegrity.lowEnergy.get_le_2_m()
```

Sets the ratio of the test packets with correct CRC transmitted by the R&S CMW. INTRO_CMD_HELP: Refer also to the following commands:

- LE RF tests (direct test mode) : Commands for LE 1M PHY - uncoded (..:LE1M..) , LE 2M PHY - uncoded (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.

- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return report_integrity: OFF | ON OFF: 100% of packets generated with correct CRC ON: 50% of packets generated with correct CRC

get_lrange() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:RINTegrity:LENergy:LRANge
value: bool = driver.configure.rxQuality.rintegrity.lowEnergy.get_lrange()
```

Sets the ratio of the test packets with correct CRC transmitted by the R&S CMW. INTRO_CMD_HELP: Refer also to the following commands:

- LE RF tests (direct test mode) : Commands for LE 1M PHY - uncoded (..:LE1M..) , LE 2M PHY - uncoded (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return report_integrity: OFF | ON OFF: 100% of packets generated with correct CRC ON: 50% of packets generated with correct CRC

set_le_1_m(*report_integrity: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:RINTegrity:LENergy[:LE1M]
driver.configure.rxQuality.rintegrity.lowEnergy.set_le_1_m(report_integrity =␣
↪False)
```

Sets the ratio of the test packets with correct CRC transmitted by the R&S CMW. INTRO_CMD_HELP: Refer also to the following commands:

- LE RF tests (direct test mode) : Commands for LE 1M PHY - uncoded (..:LE1M..) , LE 2M PHY - uncoded (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param report_integrity OFF | ON OFF: 100% of packets generated with correct CRC ON: 50% of packets generated with correct CRC

set_le_2_m(*report_integrity: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:RINTegrity:LENergy:LE2M
driver.configure.rxQuality.rintegrity.lowEnergy.set_le_2_m(report_integrity =␣
↪False)
```

Sets the ratio of the test packets with correct CRC transmitted by the R&S CMW. INTRO_CMD_HELP: Refer also to the following commands:

- LE RF tests (direct test mode) : Commands for LE 1M PHY - uncoded (..:LE1M..) , LE 2M PHY - uncoded (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param report_integrity OFF | ON OFF: 100% of packets generated with correct CRC ON: 50% of packets generated with correct CRC

set_lrange(*report_integrity: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:RINTegrity:LENergy:LRANge
driver.configure.rxQuality.rintegrity.lowEnergy.set_lrange(report_integrity =␣
↪False)
```

Sets the ratio of the test packets with correct CRC transmitted by the R&S CMW. INTRO_CMD_HELP: Refer also to the following commands:

- LE RF tests (direct test mode) : Commands for LE 1M PHY - uncoded (..:LE1M..) , LE 2M PHY - uncoded (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param report_integrity OFF | ON OFF: 100% of packets generated with correct CRC ON: 50% of packets generated with correct CRC

7.1.12.4.2 Tmode

class Tmode

Tmode commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.12.4.2.1 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:RINTegrity:TMODe:LENergy:LE1M
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:RINTegrity:TMODe:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:RINTegrity:TMODe:LENergy:LRANge
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:RINTegrity:TMODe:LENergy:LE1M
value: bool = driver.configure.rxQuality.rintegrity.tmode.lowEnergy.get_le_1_m()
```

Sets the ratio of the test packets with correct CRC transmitted by the R&S CMW. INTRO_CMD_HELP: Refer also to the following commands:

- LE RF tests (direct test mode) : Commands for LE 1M PHY - uncoded (..:LE1M..) , LE 2M PHY - uncoded (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return report_integrity: OFF | ON OFF: 100% of packets generated with correct CRC ON: 50% of packets generated with correct CRC

get_le_2_m() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:RINTegrity:TMODe:LENergy:LE2M
value: bool = driver.configure.rxQuality.rintegrity.tmode.lowEnergy.get_le_2_m()
```

Sets the ratio of the test packets with correct CRC transmitted by the R&S CMW. INTRO_CMD_HELP: Refer also to the following commands:

- LE RF tests (direct test mode) : Commands for LE 1M PHY - uncoded (..:LE1M..) , LE 2M PHY - uncoded (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return report_integrity: OFF | ON OFF: 100% of packets generated with correct CRC ON: 50% of packets generated with correct CRC

get_lrange() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:RINTegrity:TMODe:LENergy:LRANge
value: bool = driver.configure.rxQuality.rintegrity.tmode.lowEnergy.get_lrange()
```

Sets the ratio of the test packets with correct CRC transmitted by the R&S CMW. INTRO_CMD_HELP: Refer also to the following commands:

- LE RF tests (direct test mode) : Commands for LE 1M PHY - uncoded (..:LE1M..) , LE 2M PHY - uncoded (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return report_integrity: OFF | ON OFF: 100% of packets generated with correct CRC ON: 50% of packets generated with correct CRC

set_le_1_m(*report_integrity: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:RINTegrity:TMODe:LENergy:LE1M
driver.configure.rxQuality.rintegrity.tmode.lowEnergy.set_le_1_m(report_
↪integrity = False)
```

Sets the ratio of the test packets with correct CRC transmitted by the R&S CMW. INTRO_CMD_HELP: Refer also to the following commands:

- LE RF tests (direct test mode) : Commands for LE 1M PHY - uncoded (..:LE1M..) , LE 2M PHY - uncoded (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param report_integrity OFF | ON OFF: 100% of packets generated with correct CRC ON: 50% of packets generated with correct CRC

set_le_2_m(*report_integrity: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:RINTegrity:TMODe:LENergy:LE2M
driver.configure.rxQuality.rintegrity.tmode.lowEnergy.set_le_2_m(report_
↪integrity = False)
```

Sets the ratio of the test packets with correct CRC transmitted by the R&S CMW. INTRO_CMD_HELP: Refer also to the following commands:

- LE RF tests (direct test mode) : Commands for LE 1M PHY - uncoded (..:LE1M..) , LE 2M PHY - uncoded (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param report_integrity OFF | ON OFF: 100% of packets generated with correct CRC ON: 50% of packets generated with correct CRC

set_lrange(*report_integrity: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:RINTegrity:TMODe:LENergy:LRANge
driver.configure.rxQuality.rintegrity.tmode.lowEnergy.set_lrange(report_
↪integrity = False)
```

Sets the ratio of the test packets with correct CRC transmitted by the R&S CMW. INTRO_CMD_HELP: Refer also to the following commands:

- LE RF tests (direct test mode) : Commands for LE 1M PHY - uncoded (..:LE1M..) , LE 2M PHY - uncoded (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param report_integrity OFF | ON OFF: 100% of packets generated with correct CRC ON: 50% of packets generated with correct CRC

7.1.12.5 Limit

class Limit

Limit commands group definition. 14 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.12.5.1 Mper

class Mper
Mper commands group definition. 9 total commands, 3 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.12.5.1.1 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:LIMit:MPER:LENergy:LRANge
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:LIMit:MPER:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:LIMit:MPER:LENergy:LE1M
```

class LowEnergy
LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:LIMit:MPER:LENergy[:LE1M]
value: float or bool = driver.configure.rxQuality.limit.mper.lowEnergy.get_le_1_
↪m()
```

Specifies the upper PER limit for LE RX measurements. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return limit: numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default value)

get_le_2_m() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:LIMit:MPER:LENergy:LE2M
value: float or bool = driver.configure.rxQuality.limit.mper.lowEnergy.get_le_2_
↪m()
```

Specifies the upper PER limit for LE RX measurements. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return limit: numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default value)

get_lrange() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:LIMit:MPER:LENergy:LRANge
value: float or bool = driver.configure.rxQuality.limit.mper.lowEnergy.get_
↪lrange()
```

Specifies the upper PER limit for LE RX measurements. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return limit: numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default value)

set_le_1_m(*limit: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:LIMit:MPER:LENergy[:LE1M]
driver.configure.rxQuality.limit.mper.lowEnergy.set_le_1_m(limit = 1.0)
```

Specifies the upper PER limit for LE RX measurements. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param limit numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default value)

`set_le_2_m`(*limit: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:LIMit:MPER:LENergy:LE2M
driver.configure.rxQuality.limit.mper.lowEnergy.set_le_2_m(limit = 1.0)
```

Specifies the upper PER limit for LE RX measurements. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param limit numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default value)

`set_lrange`(*limit: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:LIMit:MPER:LENergy:LRANge
driver.configure.rxQuality.limit.mper.lowEnergy.set_lrange(limit = 1.0)
```

Specifies the upper PER limit for LE RX measurements. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param limit numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default value)

7.1.12.5.1.2 Tmode

`class Tmode`

Tmode commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.12.5.1.3 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:LIMit:MPER:TMODe:LENergy:LE1M
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:LIMit:MPER:TMODe:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:LIMit:MPER:TMODe:LENergy:LRANge
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:LIMit:MPER:TMODe:LENergy:LE1M
value: float or bool = driver.configure.rxQuality.limit.mper.tmode.lowEnergy.
↪get_le_1_m()
```

Specifies the upper PER limit for LE RX measurements. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return limit: numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default value)

get_le_2_m() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:LIMit:MPER:TMODe:LENergy:LE2M
value: float or bool = driver.configure.rxQuality.limit.mper.tmode.lowEnergy.
↪get_le_2_m()
```

Specifies the upper PER limit for LE RX measurements. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.

- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return limit: numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default value)

`get_lrange()` → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:LIMit:MPER:TMODe:LENergy:LRANge
value: float or bool = driver.configure.rxQuality.limit.mper.tmode.lowEnergy.
↪get_lrange()
```

Specifies the upper PER limit for LE RX measurements. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return limit: numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default value)

`set_le_1_m`(*limit: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:LIMit:MPER:TMODe:LENergy:LE1M
driver.configure.rxQuality.limit.mper.tmode.lowEnergy.set_le_1_m(limit = 1.0)
```

Specifies the upper PER limit for LE RX measurements. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param limit numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default value)

`set_le_2_m`(*limit: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:LIMit:MPER:TMODe:LENergy:LE2M
driver.configure.rxQuality.limit.mper.tmode.lowEnergy.set_le_2_m(limit = 1.0)
```

Specifies the upper PER limit for LE RX measurements. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.

- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param limit numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default value)

`set_lrange`(*limit: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:LIMit:MPER:TMODe:LENergy:LRANge
driver.configure.rxQuality.limit.mper.tmode.lowEnergy.set_lrange(limit = 1.0)
```

Specifies the upper PER limit for LE RX measurements. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param limit numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default value)

7.1.12.5.1.4 Nmode

class Nmode

Nmode commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.12.5.1.5 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:LIMit:MPER:NMODe:LENergy:LE1M
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:LIMit:MPER:NMODe:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:LIMit:MPER:NMODe:LENergy:LRANge
```

class LowEnergy

LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

`get_le_1_m`() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:LIMit:MPER:NMODe:LENergy:LE1M
value: float or bool = driver.configure.rxQuality.limit.mper.nmode.lowEnergy.
↪get_le_1_m()
```

Specifies the upper PER limit for LE RX measurements. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return limit: numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default value)

get_le_2_m() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:LIMit:MPER:NMODe:LENergy:LE2M
value: float or bool = driver.configure.rxQuality.limit.mper.nmode.lowEnergy.
↪get_le_2_m()
```

Specifies the upper PER limit for LE RX measurements. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return limit: numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default value)

get_lrange() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:LIMit:MPER:NMODe:LENergy:LRANge
value: float or bool = driver.configure.rxQuality.limit.mper.nmode.lowEnergy.
↪get_lrange()
```

Specifies the upper PER limit for LE RX measurements. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

return limit: numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default value)

set_le_1_m(*limit: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:LIMit:MPER:NMODe:LENergy:LE1M
driver.configure.rxQuality.limit.mper.nmode.lowEnergy.set_le_1_m(limit = 1.0)
```

Specifies the upper PER limit for LE RX measurements. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param limit numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default value)

set_le_2_m(*limit: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:LIMit:MPER:NMODe:LENergy:LE2M
driver.configure.rxQuality.limit.mper.nmode.lowEnergy.set_le_2_m(limit = 1.0)
```

Specifies the upper PER limit for LE RX measurements. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param limit numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default value)

set_lrange(*limit: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:LIMit:MPER:NMODe:LENergy:LRANge
driver.configure.rxQuality.limit.mper.nmode.lowEnergy.set_lrange(limit = 1.0)
```

Specifies the upper PER limit for LE RX measurements. INTRO_CMD_HELP: Refer also to the following commands:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands ..:NMODe:LENergy:.. are available.
- LE test mode: Commands ..:TMODe:LENergy:.. are available.

param limit numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default value)

7.1.12.5.2 Mber

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:LIMit:MBER:BRATe
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:LIMit:MBER:EDRate
```

class Mber

Mber commands group definition. 5 total commands, 1 Sub-groups, 2 group commands

get_brate() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:LIMit:MBER:BRATe
value: float or bool = driver.configure.rxQuality.limit.mber.get_brate()
```

Specifies the upper BER limit for BR RX measurements.

return limit: numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default level)

get_edrate() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:LIMit:MBER:EDRate
value: float or bool = driver.configure.rxQuality.limit.mber.get_edrate()
```

Specifies the upper BER limit for EDR RX measurements.

return limit: numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default level)

set_brate(*limit: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:LIMit:MBER:BRATe
driver.configure.rxQuality.limit.mber.set_brate(limit = 1.0)
```

Specifies the upper BER limit for BR RX measurements.

param limit numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default level)

set_edrate(*limit: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:LIMit:MBER:EDRate
driver.configure.rxQuality.limit.mber.set_edrate(limit = 1.0)
```

Specifies the upper BER limit for EDR RX measurements.

param limit numeric | ON | OFF Range: 0 % to 100 %, Unit: % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default level)

Cloning the Group

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

Subgroups

7.1.12.5.2.1 Tmode

class Tmode
Tmode commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.12.5.2.2 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:LIMit:MBER:TMODe:LENergy:LE1M
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:LIMit:MBER:TMODe:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:LIMit:MBER:TMODe:LENergy:LRANge
```

class LowEnergy
LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_le_1_m() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:LIMit:MBER:TMODe:LENergy:LE1M
value: float or bool = driver.configure.rxQuality.limit.mber.tmode.lowEnergy.
↪get_le_1_m()
```

Specifies the upper BER limit for PER measurements in LE test mode. Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M. .) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (..:TMODe:LENergy:LRANge..) are available.

return limit: numeric | ON | OFF Range: 0 % to 100 % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default value)

get_le_2_m() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:LIMit:MBER:TMODe:LENergy:LE2M
value: float or bool = driver.configure.rxQuality.limit.mber.tmode.lowEnergy.
↪get_le_2_m()
```

(continues on next page)

(continued from previous page)

Specifies the upper BER limit for PER measurements in LE test mode. Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M. .) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (..:TMODe:LENergy:LRANge..) are available.

return limit: numeric | ON | OFF Range: 0 % to 100 % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default value)

get_lrange() → float

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:LIMit:MBER:TMODe:LENergy:LRANge
value: float or bool = driver.configure.rxQuality.limit.mber.tmode.lowEnergy.
↪get_lrange()
```

Specifies the upper BER limit for PER measurements in LE test mode. Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M. .) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (..:TMODe:LENergy:LRANge..) are available.

return limit: numeric | ON | OFF Range: 0 % to 100 % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default value)

set_le_1_m(*limit: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:LIMit:MBER:TMODe:LENergy:LE1M
driver.configure.rxQuality.limit.mber.tmode.lowEnergy.set_le_1_m(limit = 1.0)
```

Specifies the upper BER limit for PER measurements in LE test mode. Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M. .) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (..:TMODe:LENergy:LRANge..) are available.

param limit numeric | ON | OFF Range: 0 % to 100 % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default value)

set_le_2_m(*limit: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:LIMit:MBER:TMODe:LENergy:LE2M
driver.configure.rxQuality.limit.mber.tmode.lowEnergy.set_le_2_m(limit = 1.0)
```

Specifies the upper BER limit for PER measurements in LE test mode. Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M. .) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (..:TMODe:LENergy:LRANge..) are available.

param limit numeric | ON | OFF Range: 0 % to 100 % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default value)

set_lrange(*limit: float*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:LIMit:MBER:TMODe:LENergy:LRANge
driver.configure.rxQuality.limit.mber.tmode.lowEnergy.set_lrange(limit = 1.0)
```

Specifies the upper BER limit for PER measurements in LE test mode. Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M. .) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (..:TMODe:LENergy:LRANge..) are available.

param limit numeric | ON | OFF Range: 0 % to 100 % Additional parameters: OFF | ON (disables the limit | enables the limit using the previous/default value)

7.1.12.6 IbLength

class IbLength

IbLength commands group definition. 2 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.12.6.1 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:IBLength:LENergy:LE1M
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:IBLength:LENergy:LE2M
```

class LowEnergy

LowEnergy commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

class Le1MStruct

Structure for reading output parameters. Fields:

- Enable: bool: OFF | ON
- Inter_Burst_Len: int: integer Range: 0 slot(s) to 255 slot(s)

class Le2MStruct

Structure for reading output parameters. Fields:

- Enable: bool: OFF | ON
- Inter_Burst_Len: int: integer Range: 0 slot(s) to 255 slot(s)

get_le_1_m() → Le1MStruct

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:IBLength:LENergy:LE1M
value: Le1MStruct = driver.configure.rxQuality.ibLength.lowEnergy.get_le_1_m()
```

Enables and sets the number of slots to wait between transmissions for direction finding tests.

return structure: for return value, see the help for Le1MStruct structure arguments.

get_le_2_m() → Le2MStruct

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:IBLength:LENergy:LE2M
value: Le2MStruct = driver.configure.rxQuality.ibLength.lowEnergy.get_le_2_m()
```

Enables and sets the number of slots to wait between transmissions for direction finding tests.

return structure: for return value, see the help for Le2MStruct structure arguments.

set_le_1_m(*value: RsCmwBluetoothSig.Implementations.Configure_.RxQuality_.IbLength_.LowEnergy.LowEnergy.Le1MStruct*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:IBLength:LENergy:LE1M
driver.configure.rxQuality.ibLength.lowEnergy.set_le_1_m(value = Le1MStruct())
```

Enables and sets the number of slots to wait between transmissions for direction finding tests.

param value see the help for Le1MStruct structure arguments.

set_le_2_m(*value: RsCmwBluetoothSig.Implementations.Configure_.RxQuality_.IbLength_.LowEnergy.LowEnergy.Le2MStruct*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:IBLength:LENergy:LE2M
driver.configure.rxQuality.ibLength.lowEnergy.set_le_2_m(value = Le2MStruct())
```

Enables and sets the number of slots to wait between transmissions for direction finding tests.

param value see the help for Le2MStruct structure arguments.

7.1.12.7 IqCoherency

class IqCoherency

IqCoherency commands group definition. 20 total commands, 4 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.12.7.1 MoException

class MoException

MoException commands group definition. 2 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.12.7.1.1 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:IQCoherency:MOEXception:LENergy:LE1M
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:IQCoherency:MOEXception:LENergy:LE2M
```

class LowEnergy

LowEnergy commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

get_le_1_m() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:MOEXception:LENergy:LE1M
value: bool = driver.configure.rxQuality.iqCoherency.moException.lowEnergy.get_
↪le_1_m()
```

Specifies whether measurement results that the R&S CMW identifies as faulty or inaccurate are rejected. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

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

get_le_2_m() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:MOEXception:LENergy:LE2M
value: bool = driver.configure.rxQuality.iqCoherency.moException.lowEnergy.get_
↪le_2_m()
```

Specifies whether measurement results that the R&S CMW identifies as faulty or inaccurate are rejected. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

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

set_le_1_m(*meas_on_exception: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:MOEXception:LENergy:LE1M
driver.configure.rxQuality.iqCoherency.moException.lowEnergy.set_le_1_m(meas_on_
↪exception = False)
```

Specifies whether measurement results that the R&S CMW identifies as faulty or inaccurate are rejected. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

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

set_le_2_m(*meas_on_exception: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:MOEXception:LENergy:LE2M
driver.configure.rxQuality.iqCoherency.moException.lowEnergy.set_le_2_m(meas_on_
↪exception = False)
```

Specifies whether measurement results that the R&S CMW identifies as faulty or inaccurate are rejected. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

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

7.1.12.7.2 NoMeas

class NoMeas

NoMeas commands group definition. 8 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.12.7.2.1 LowEnergy

class LowEnergy

LowEnergy commands group definition. 8 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.12.7.2.2 Le1M

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:NOMeas:LENergy:LE1M:A0Reference
CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:NOMeas:LENergy:LE1M:A1NReference
CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:NOMeas:LENergy:LE1M:A2NReference
CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:NOMeas:LENergy:LE1M:A3NReference
```

class Le1M

Le1M commands group definition. 4 total commands, 0 Sub-groups, 4 group commands

get_a_0_reference() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:NOMeas:LENergy:LE1M:A0Reference
value: int = driver.configure.rxQuality.iqCoherency.noMeas.lowEnergy.le1M.get_a_
↪0_reference()
```

Specifies the minimal number of IQ samples to be reported by the EUT for the specified antenna. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second antenna (...:A1NReference) , and optional third and fourth antennas are available.

return no_of_meas: numeric Range: 1 to 30E+3

get_a_1_nreference() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:NOMeas:LENergy:LE1M:A1NReference
value: int = driver.configure.rxQuality.iqCoherency.noMeas.lowEnergy.le1M.get_a_
↪1_nreference()
```

Specifies the minimal number of IQ samples to be reported by the EUT for the specified antenna. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second antenna (...:A1NReference) , and optional third and fourth antennas are available.

return no_of_meas: numeric Range: 1 to 30E+3

get_a_2_nreference() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:NOMeas:LENergy:LE1M:A2NReference
value: int = driver.configure.rxQuality.iqCoherency.noMeas.lowEnergy.le1M.get_a_
↪2_nreference()
```

Specifies the minimal number of IQ samples to be reported by the EUT for the specified antenna. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second antenna (...:A1NReference) , and optional third and fourth antennas are available.

return no_of_meas: numeric Range: 1 to 30E+3

get_a_3_nreference() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:NOMeas:LENergy:LE1M:A3NReference
value: int = driver.configure.rxQuality.iqCoherency.noMeas.lowEnergy.le1M.get_a_
↪3_nreference()
```

Specifies the minimal number of IQ samples to be reported by the EUT for the specified antenna. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second antenna (...:A1NReference) , and optional third and fourth antennas are available.

return no_of_meas: numeric Range: 1 to 30E+3

set_a_0_reference(*no_of_meas: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:NOMeas:LENergy:LE1M:A0Reference
driver.configure.rxQuality.iqCoherency.noMeas.lowEnergy.le1M.set_a_0_
↪reference(no_of_meas = 1)
```

Specifies the minimal number of IQ samples to be reported by the EUT for the specified antenna. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second antenna (...:A1NReference) , and optional third and fourth antennas are available.

param no_of_meas numeric Range: 1 to 30E+3

set_a_1_nreference(*no_of_meas: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:NOMeas:LENergy:LE1M:A1NReference
driver.configure.rxQuality.iqCoherency.noMeas.lowEnergy.le1M.set_a_1_
↪nreference(no_of_meas = 1)
```

Specifies the minimal number of IQ samples to be reported by the EUT for the specified antenna. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second antenna (...:A1NReference) , and optional third and fourth antennas are available.

param no_of_meas numeric Range: 1 to 30E+3

set_a_2_nreference(*no_of_meas: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:NOMeas:LENergy:LE1M:A2NReference
driver.configure.rxQuality.iqCoherency.noMeas.lowEnergy.le1M.set_a_2_
↪nreference(no_of_meas = 1)
```

Specifies the minimal number of IQ samples to be reported by the EUT for the specified antenna. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second antenna (...:A1NReference) , and optional third and fourth antennas are available.

param no_of_meas numeric Range: 1 to 30E+3

set_a_3_nreference(*no_of_meas: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:NOMeas:LENergy:LE1M:A3NReference
driver.configure.rxQuality.iqCoherency.noMeas.lowEnergy.le1M.set_a_3_
↪nreference(no_of_meas = 1)
```

Specifies the minimal number of IQ samples to be reported by the EUT for the specified antenna. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second antenna (...:A1NReference) , and optional third and fourth antennas are available.

param no_of_meas numeric Range: 1 to 30E+3

7.1.12.7.2.3 Le2M

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:NOMeas:LENergy:LE2M:A0Reference
CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:NOMeas:LENergy:LE2M:A1NReference
CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:NOMeas:LENergy:LE2M:A2NReference
CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:NOMeas:LENergy:LE2M:A3NReference
```

class Le2M

Le2M commands group definition. 4 total commands, 0 Sub-groups, 4 group commands

get_a_0_reference() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:NOMeas:LENergy:LE2M:A0Reference
value: int = driver.configure.rxQuality.iqCoherency.noMeas.lowEnergy.le2M.get_a_
↪0_reference()
```

Specifies the minimal number of IQ samples to be reported by the EUT for the specified antenna. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second antenna (...:A1NReference) , and optional third and fourth antennas are available.

return no_of_meas: numeric Range: 1 to 30E+3

get_a_1_nreference() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:NOMeas:LENergy:LE2M:A1NReference
value: int = driver.configure.rxQuality.iqCoherency.noMeas.lowEnergy.le2M.get_a_
↪1_nreference()
```

Specifies the minimal number of IQ samples to be reported by the EUT for the specified antenna. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second antenna (...:A1NReference) , and optional third and fourth antennas are available.

return no_of_meas: numeric Range: 1 to 30E+3

get_a_2_nreference() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:NOMeas:LENergy:LE2M:A2NReference
value: int = driver.configure.rxQuality.iqCoherency.noMeas.lowEnergy.le2M.get_a_
↪2_nreference()
```

Specifies the minimal number of IQ samples to be reported by the EUT for the specified antenna. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second antenna (...:A1NReference) , and optional third and fourth antennas are available.

return no_of_meas: numeric Range: 1 to 30E+3

get_a_3_nreference() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:NOMeas:LENergy:LE2M:A3NReference
value: int = driver.configure.rxQuality.iqCoherency.noMeas.lowEnergy.le2M.get_a_
↪3_nreference()
```

Specifies the minimal number of IQ samples to be reported by the EUT for the specified antenna. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second antenna (...:A1NReference) , and optional third and fourth antennas are available.

return no_of_meas: numeric Range: 1 to 30E+3

set_a_0_reference(*no_of_meas: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:NOMeas:LENergy:LE2M:A0Reference
driver.configure.rxQuality.iqCoherency.noMeas.lowEnergy.le2M.set_a_0_
↪reference(no_of_meas = 1)
```

Specifies the minimal number of IQ samples to be reported by the EUT for the specified antenna. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second antenna (...:A1NReference) , and optional third and fourth antennas are available.

param no_of_meas numeric Range: 1 to 30E+3

set_a_1_nreference(*no_of_meas: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:NOMeas:LENergy:LE2M:A1NReference
driver.configure.rxQuality.iqCoherency.noMeas.lowEnergy.le2M.set_a_1_
↪nreference(no_of_meas = 1)
```

(continues on next page)

(continued from previous page)

Specifies the minimal number of IQ samples to be reported by the EUT for the specified antenna. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second antenna (...:A1NReference) , and optional third and fourth antennas are available.

param no_of_meas numeric Range: 1 to 30E+3

set_a_2_nreference(*no_of_meas: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:NOMeas:LENergy:LE2M:A2NReference
driver.configure.rxQuality.iqCoherency.noMeas.lowEnergy.le2M.set_a_2_
↪nreference(no_of_meas = 1)
```

Specifies the minimal number of IQ samples to be reported by the EUT for the specified antenna. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second antenna (...:A1NReference) , and optional third and fourth antennas are available.

param no_of_meas numeric Range: 1 to 30E+3

set_a_3_nreference(*no_of_meas: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:NOMeas:LENergy:LE2M:A3NReference
driver.configure.rxQuality.iqCoherency.noMeas.lowEnergy.le2M.set_a_3_
↪nreference(no_of_meas = 1)
```

Specifies the minimal number of IQ samples to be reported by the EUT for the specified antenna. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second antenna (...:A1NReference) , and optional third and fourth antennas are available.

param no_of_meas numeric Range: 1 to 30E+3

7.1.12.7.3 Packets

class Packets

Packets commands group definition. 2 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.12.7.3.1 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:IQCoherency:PACKets:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:IQCoherency:PACKets:LENergy:LE1M
```

class LowEnergy

LowEnergy commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

get_le_1_m() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:PACKets:LENergy:LE1M
value: int = driver.configure.rxQuality.iqCoherency.packets.lowEnergy.get_le_1_
↪m()
```

Defines the number of packets to be sent per measurement cycle (statistics cycle) . Commands for uncoded LE 1M PHY (.. :LE1M..) and LE 2M PHY (..:LE2M..) are available.

return packets: numeric Range: 1 to 6.4E+6

get_le_2_m() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:PACKets:LENergy:LE2M
value: int = driver.configure.rxQuality.iqCoherency.packets.lowEnergy.get_le_2_
↪m()
```

Defines the number of packets to be sent per measurement cycle (statistics cycle) . Commands for uncoded LE 1M PHY (.. :LE1M..) and LE 2M PHY (..:LE2M..) are available.

return packets: numeric Range: 1 to 6.4E+6

set_le_1_m(*packets: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:PACKets:LENergy:LE1M
driver.configure.rxQuality.iqCoherency.packets.lowEnergy.set_le_1_m(packets = 1)
```

Defines the number of packets to be sent per measurement cycle (statistics cycle) . Commands for uncoded LE 1M PHY (.. :LE1M..) and LE 2M PHY (..:LE2M..) are available.

param packets numeric Range: 1 to 6.4E+6

set_le_2_m(*packets: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:PACKets:LENergy:LE2M
driver.configure.rxQuality.iqCoherency.packets.lowEnergy.set_le_2_m(packets = 1)
```

Defines the number of packets to be sent per measurement cycle (statistics cycle) . Commands for uncoded LE 1M PHY (.. :LE1M..) and LE 2M PHY (..:LE2M..) are available.

param packets numeric Range: 1 to 6.4E+6

7.1.12.7.4 Limit

class Limit
Limit commands group definition. 8 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.12.7.4.1 LowEnergy

class LowEnergy
LowEnergy commands group definition. 8 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.12.7.4.2 Le1M

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LIMit:LENergy:LE1M:A0Reference
CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LIMit:LENergy:LE1M:A1NReference
CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LIMit:LENergy:LE1M:A2NReference
CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LIMit:LENergy:LE1M:A3NReference
```

class Le1M
Le1M commands group definition. 4 total commands, 0 Sub-groups, 4 group commands

class A0ReferenceStruct
Structure for reading output parameters. Fields:

- Limit: float or bool: numeric Range: 0 (Rad) to 3.14 (Rad)

- Enable: bool: OFF | ON Disables/enables the limit check

class A1NreferenceStruct
Structure for reading output parameters. Fields:

- Limit: float or bool: numeric Range: 0 (Rad) to 3.14 (Rad)
- Enable: bool: OFF | ON Disables/enables the limit check

class A2NreferenceStruct
Structure for reading output parameters. Fields:

- Limit: float or bool: numeric Range: 0 (Rad) to 3.14 (Rad)
- Enable: bool: OFF | ON Disables/enables the limit check

class A3NreferenceStruct
Structure for reading output parameters. Fields:

- Limit: float or bool: numeric Range: 0 (Rad) to 3.14 (Rad)
- Enable: bool: OFF | ON Disables/enables the limit check

get_a_0_reference() → A0ReferenceStruct

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LIMit:LENergy:LE1M:A0Reference
value: A0ReferenceStruct = driver.configure.rxQuality.iqCoherency.limit.
↪lowEnergy.le1M.get_a_0_reference()
```

Defines the IQ samples coherency limit for mean reference phase deviation (RPD) results for the reference antenna. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

return structure: for return value, see the help for A0ReferenceStruct structure arguments.

get_a_1_nreference() → A1NreferenceStruct

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LIMit:LENergy:LE1M:A1NReference
value: A1NreferenceStruct = driver.configure.rxQuality.iqCoherency.limit.
↪lowEnergy.le1M.get_a_1_nreference()
```

Defines the IQ samples coherency limit for 95% relative phase values RP(m) for non-reference antennas A1NReference to A3NReference. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

return structure: for return value, see the help for A1NreferenceStruct structure arguments.

get_a_2_nreference() → A2NreferenceStruct

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LIMit:LENergy:LE1M:A2NReference
value: A2NreferenceStruct = driver.configure.rxQuality.iqCoherency.limit.
↪lowEnergy.le1M.get_a_2_nreference()
```

Defines the IQ samples coherency limit for 95% relative phase values RP(m) for non-reference antennas A1NReference to A3NReference. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

return structure: for return value, see the help for A2NreferenceStruct structure arguments.

get_a_3_nreference() → A3NreferenceStruct

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LIMit:LENergy:LE1M:A3NReference
value: A3NreferenceStruct = driver.configure.rxQuality.iqCoherency.limit.
↪lowEnergy.le1M.get_a_3_nreference()
```

Defines the IQ samples coherency limit for 95% relative phase values RP(m) for non-reference antennas A1NReference to A3NReference. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

return structure: for return value, see the help for A3NreferenceStruct structure arguments.

set_a_0_reference(*value: RsCmwBluetooth-Sig.Implementations.Configure_.RxQuality_.IqCoherency_.Limit_.LowEnergy_.Le1M.Le1M.A0Reference*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LIMit:LENergy:LE1M:A0Reference
driver.configure.rxQuality.iqCoherency.limit.lowEnergy.le1M.set_a_0_
↪reference(value = A0ReferenceStruct())
```

Defines the IQ samples coherency limit for mean reference phase deviation (RPD) results for the reference antenna. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

param value see the help for A0ReferenceStruct structure arguments.

set_a_1_nreference(*value: RsCmwBluetooth-Sig.Implementations.Configure_.RxQuality_.IqCoherency_.Limit_.LowEnergy_.Le1M.Le1M.A1Nreferen*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LIMit:LENergy:LE1M:A1NReference
driver.configure.rxQuality.iqCoherency.limit.lowEnergy.le1M.set_a_1_
↪nreference(value = A1NreferenceStruct())
```

Defines the IQ samples coherency limit for 95% relative phase values RP(m) for non-reference antennas A1NReference to A3NReference. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

param value see the help for A1NreferenceStruct structure arguments.

set_a_2_nreference(*value: RsCmwBluetooth-Sig.Implementations.Configure_.RxQuality_.IqCoherency_.Limit_.LowEnergy_.Le1M.Le1M.A2Nreferen*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LIMit:LENergy:LE1M:A2NReference
driver.configure.rxQuality.iqCoherency.limit.lowEnergy.le1M.set_a_2_
↪nreference(value = A2NreferenceStruct())
```

Defines the IQ samples coherency limit for 95% relative phase values RP(m) for non-reference antennas A1NReference to A3NReference. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

param value see the help for A2NreferenceStruct structure arguments.

set_a_3_nreference(*value: RsCmwBluetooth-Sig.Implementations.Configure_.RxQuality_.IqCoherency_.Limit_.LowEnergy_.Le1M.Le1M.A3Nreferen*
→ None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LIMit:LENergy:LE1M:A3NReference
driver.configure.rxQuality.iqCoherency.limit.lowEnergy.le1M.set_a_3_
↪nreference(value = A3NreferenceStruct())
```

Defines the IQ samples coherency limit for 95% relative phase values RP(m) for non-reference antennas A1NReference to A3NReference. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

param value see the help for A3NreferenceStruct structure arguments.

7.1.12.7.4.3 Le2M

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LIMit:LENergy:LE2M:A0Reference
CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LIMit:LENergy:LE2M:A1NReference
CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LIMit:LENergy:LE2M:A2NReference
CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LIMit:LENergy:LE2M:A3NReference
```

class Le2M

Le2M commands group definition. 4 total commands, 0 Sub-groups, 4 group commands

class A0ReferenceStruct

Structure for reading output parameters. Fields:

- Limit: float or bool: numeric Range: 0 (Rad) to 3.14 (Rad)
- Enable: bool: OFF | ON Disables/enables the limit check

class A1NreferenceStruct

Structure for reading output parameters. Fields:

- Limit: float or bool: numeric Range: 0 (Rad) to 3.14 (Rad)
- Enable: bool: OFF | ON Disables/enables the limit check

class A2NreferenceStruct
Structure for reading output parameters. Fields:

- Limit: float or bool: numeric Range: 0 (Rad) to 3.14 (Rad)
- Enable: bool: OFF | ON Disables/enables the limit check

class A3NreferenceStruct
Structure for reading output parameters. Fields:

- Limit: float or bool: numeric Range: 0 (Rad) to 3.14 (Rad)
- Enable: bool: OFF | ON Disables/enables the limit check

get_a_0_reference() → A0ReferenceStruct

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LIMit:LENergy:LE2M:A0Reference
value: A0ReferenceStruct = driver.configure.rxQuality.iqCoherency.limit.
↪lowEnergy.le2M.get_a_0_reference()
```

Defines the IQ samples coherency limit for mean reference phase deviation (RPD) results for the reference antenna. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

return structure: for return value, see the help for A0ReferenceStruct structure arguments.

get_a_1_nreference() → A1NreferenceStruct

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LIMit:LENergy:LE2M:A1NReference
value: A1NreferenceStruct = driver.configure.rxQuality.iqCoherency.limit.
↪lowEnergy.le2M.get_a_1_nreference()
```

Defines the IQ samples coherency limit for 95% relative phase values RP(m) for non-reference antennas A1NReference to A3NReference. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

return structure: for return value, see the help for A1NreferenceStruct structure arguments.

get_a_2_nreference() → A2NreferenceStruct

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LIMit:LENergy:LE2M:A2NReference
value: A2NreferenceStruct = driver.configure.rxQuality.iqCoherency.limit.
↪lowEnergy.le2M.get_a_2_nreference()
```

Defines the IQ samples coherency limit for 95% relative phase values RP(m) for non-reference antennas A1NReference to A3NReference. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

return structure: for return value, see the help for A2NreferenceStruct structure arguments.

get_a_3_nreference() → A3NreferenceStruct

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LIMit:LENergy:LE2M:A3NReference
value: A3NreferenceStruct = driver.configure.rxQuality.iqCoherency.limit.
↪lowEnergy.le2M.get_a_3_nreference()
```

Defines the IQ samples coherency limit for 95% relative phase values RP(m) for non-reference antennas A1NReference to A3NReference. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

return structure: for return value, see the help for A3NreferenceStruct structure arguments.

set_a_0_reference(*value: RsCmwBluetoothSig.Implementations.Configure_.RxQuality_.IqCoherency_.Limit_.LowEnergy_.Le2M.Le2M.A0Reference*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LIMit:LENergy:LE2M:A0Reference
driver.configure.rxQuality.iqCoherency.limit.lowEnergy.le2M.set_a_0_
↪reference(value = A0ReferenceStruct())
```

Defines the IQ samples coherency limit for mean reference phase deviation (RPD) results for the reference antenna. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

param value see the help for A0ReferenceStruct structure arguments.

set_a_1_nreference(*value: RsCmwBluetoothSig.Implementations.Configure_.RxQuality_.IqCoherency_.Limit_.LowEnergy_.Le2M.Le2M.A1Nreferen*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LIMit:LENergy:LE2M:A1NReference
driver.configure.rxQuality.iqCoherency.limit.lowEnergy.le2M.set_a_1_
↪nreference(value = A1NreferenceStruct())
```

Defines the IQ samples coherency limit for 95% relative phase values RP(m) for non-reference antennas A1NReference to A3NReference. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

param value see the help for A1NreferenceStruct structure arguments.

set_a_2_nreference(*value: RsCmwBluetoothSig.Implementations.Configure_.RxQuality_.IqCoherency_.Limit_.LowEnergy_.Le2M.Le2M.A2Nreferen*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LIMit:LENergy:LE2M:A2NReference
driver.configure.rxQuality.iqCoherency.limit.lowEnergy.le2M.set_a_2_
↪nreference(value = A2NreferenceStruct())
```

Defines the IQ samples coherency limit for 95% relative phase values RP(m) for non-reference antennas A1NReference to A3NReference. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

param value see the help for A2NreferenceStruct structure arguments.

set_a_3_nreference(*value: RsCmwBluetooth-Sig.Implementations.Configure_.RxQuality_.IqCoherency_.Limit_.LowEnergy_.Le2M.Le2M.A3Nreferen*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LIMit:LENergy:LE2M:A3NReference
driver.configure.rxQuality.iqCoherency.limit.lowEnergy.le2M.set_a_3_
↪nreference(value = A3NreferenceStruct())
```

Defines the IQ samples coherency limit for 95% relative phase values RP(m) for non-reference antennas A1NReference to A3NReference. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

param value see the help for A3NreferenceStruct structure arguments.

7.1.12.8 IqDrange

class IqDrange

IqDrange commands group definition. 16 total commands, 5 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.12.8.1 MoException

class MoException

MoException commands group definition. 2 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.12.8.1.1 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange:MOEXception:LENergy:LE1M
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange:MOEXception:LENergy:LE2M
```

class LowEnergy

LowEnergy commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

get_le_1_m() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:MOEXception:LENergy:LE1M
value: bool = driver.configure.rxQuality.iqDrange.moException.lowEnergy.get_le_
↪1_m()
```

Specifies whether measurement results that the R&S CMW identifies as faulty or inaccurate are rejected. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

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

get_le_2_m() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:MOEXception:LENergy:LE2M
value: bool = driver.configure.rxQuality.iqDrange.moException.lowEnergy.get_le_
↪2_m()
```

Specifies whether measurement results that the R&S CMW identifies as faulty or inaccurate are rejected. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

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

set_le_1_m(*meas_on_exception: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:MOEXception:LENergy:LE1M
driver.configure.rxQuality.iqDrange.moException.lowEnergy.set_le_1_m(meas_on_
↪exception = False)
```

Specifies whether measurement results that the R&S CMW identifies as faulty or inaccurate are rejected. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

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

set_le_2_m(*meas_on_exception: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:MOEXception:LENergy:LE2M
driver.configure.rxQuality.iqDrange.moException.lowEnergy.set_le_2_m(meas_on_
↪exception = False)
```

Specifies whether measurement results that the R&S CMW identifies as faulty or inaccurate are rejected. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available.

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

7.1.12.8.2 NoMeas

class NoMeas
NoMeas commands group definition. 8 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.12.8.2.1 LowEnergy

class LowEnergy
LowEnergy commands group definition. 8 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.12.8.2.2 Le1M

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:NOMeas:LENergy:LE1M:A0Reference
CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:NOMeas:LENergy:LE1M:A1NReference
CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:NOMeas:LENergy:LE1M:A2NReference
CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:NOMeas:LENergy:LE1M:A3NReference
```

class Le1M
Le1M commands group definition. 4 total commands, 0 Sub-groups, 4 group commands

get_a_0_reference() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:NOMeas:LENergy:LE1M:A0Reference
value: int = driver.configure.rxQuality.iqDrange.noMeas.lowEnergy.le1M.get_a_0_
↪reference()
```

Defines the number of measurements per measurement cycle for the specified antenna. Commands for uncoded LE 1M PHY (.. :LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second antenna (...:A1NReference) , and optional third and fourth antennas are available.

return no_of_meas: numeric Range: 0 to 30E+3

get_a_1_nreference() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:NOMeas:LENergy:LE1M:A1NReference
value: int = driver.configure.rxQuality.iqDrange.noMeas.lowEnergy.le1M.get_a_1_
↪nreference()
```

Defines the number of measurements per measurement cycle for the specified antenna. Commands for uncoded LE 1M PHY (.. :LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second antenna (...:A1NReference) , and optional third and fourth antennas are available.

return no_of_meas: numeric Range: 0 to 30E+3

get_a_2_nreference() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:NOMeas:LENergy:LE1M:A2NReference
value: int = driver.configure.rxQuality.iqDrange.noMeas.lowEnergy.le1M.get_a_2_
↪nreference()
```

Defines the number of measurements per measurement cycle for the specified antenna. Commands for uncoded LE 1M PHY (.. :LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second antenna (...:A1NReference) , and optional third and fourth antennas are available.

return no_of_meas: numeric Range: 0 to 30E+3

get_a_3_nreference() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:NOMeas:LENergy:LE1M:A3NReference
value: int = driver.configure.rxQuality.iqDrange.noMeas.lowEnergy.le1M.get_a_3_
↪nreference()
```

Defines the number of measurements per measurement cycle for the specified antenna. Commands for uncoded LE 1M PHY (.. :LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second antenna (...:A1NReference) , and optional third and fourth antennas are available.

return no_of_meas: numeric Range: 0 to 30E+3

set_a_0_reference(*no_of_meas: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:NOMeas:LENergy:LE1M:A0Reference
driver.configure.rxQuality.iqDrange.noMeas.lowEnergy.le1M.set_a_0_reference(no_
↪of_meas = 1)
```

(continues on next page)

(continued from previous page)

Defines the number of measurements per measurement cycle for the specified antenna. Commands for uncoded LE 1M PHY (.. :LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second antenna (...:A1NReference) , and optional third and fourth antennas are available.

param no_of_meas numeric Range: 0 to 30E+3

set_a_1_nreference(*no_of_meas: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:NOMeas:LENergy:LE1M:A1NReference
driver.configure.rxQuality.iqDrange.noMeas.lowEnergy.le1M.set_a_1_nreference(no_
↪of_meas = 1)
```

Defines the number of measurements per measurement cycle for the specified antenna. Commands for uncoded LE 1M PHY (.. :LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second antenna (...:A1NReference) , and optional third and fourth antennas are available.

param no_of_meas numeric Range: 0 to 30E+3

set_a_2_nreference(*no_of_meas: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:NOMeas:LENergy:LE1M:A2NReference
driver.configure.rxQuality.iqDrange.noMeas.lowEnergy.le1M.set_a_2_nreference(no_
↪of_meas = 1)
```

Defines the number of measurements per measurement cycle for the specified antenna. Commands for uncoded LE 1M PHY (.. :LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second antenna (...:A1NReference) , and optional third and fourth antennas are available.

param no_of_meas numeric Range: 0 to 30E+3

set_a_3_nreference(*no_of_meas: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:NOMeas:LENergy:LE1M:A3NReference
driver.configure.rxQuality.iqDrange.noMeas.lowEnergy.le1M.set_a_3_nreference(no_
↪of_meas = 1)
```

Defines the number of measurements per measurement cycle for the specified antenna. Commands for uncoded LE 1M PHY (.. :LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second antenna (...:A1NReference) , and optional third and fourth antennas are available.

param no_of_meas numeric Range: 0 to 30E+3

7.1.12.8.2.3 Le2M

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:NOMeas:LENergy:LE2M:A0Reference
CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:NOMeas:LENergy:LE2M:A1NReference
CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:NOMeas:LENergy:LE2M:A2NReference
CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:NOMeas:LENergy:LE2M:A3NReference
```

class Le2M

Le2M commands group definition. 4 total commands, 0 Sub-groups, 4 group commands

get_a_0_reference() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:NOMeas:LENergy:LE2M:A0Reference
value: int = driver.configure.rxQuality.iqDrange.noMeas.lowEnergy.le2M.get_a_0_
↪reference()
```

Defines the number of measurements per measurement cycle for the specified antenna. Commands for uncoded LE 1M PHY (.. :LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second antenna (...:A1NReference) , and optional third and fourth antennas are available.

return no_of_meas: numeric Range: 0 to 30E+3

get_a_1_nreference() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:NOMeas:LENergy:LE2M:A1NReference
value: int = driver.configure.rxQuality.iqDrange.noMeas.lowEnergy.le2M.get_a_1_
↪nreference()
```

Defines the number of measurements per measurement cycle for the specified antenna. Commands for uncoded LE 1M PHY (.. :LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second antenna (...:A1NReference) , and optional third and fourth antennas are available.

return no_of_meas: numeric Range: 0 to 30E+3

get_a_2_nreference() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:NOMeas:LENergy:LE2M:A2NReference
value: int = driver.configure.rxQuality.iqDrange.noMeas.lowEnergy.le2M.get_a_2_
↪nreference()
```

Defines the number of measurements per measurement cycle for the specified antenna. Commands for uncoded LE 1M PHY (.. :LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for reference

antenna (...:A0Reference) , mandatory second antenna (...:A1NReference) , and optional third and fourth antennas are available.

return no_of_meas: numeric Range: 0 to 30E+3

get_a_3_nreference() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:NOMeas:LENergy:LE2M:A3NReference
value: int = driver.configure.rxQuality.iqDrange.noMeas.lowEnergy.le2M.get_a_3_
↪nreference()
```

Defines the number of measurements per measurement cycle for the specified antenna. Commands for uncoded LE 1M PHY (.. :LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second antenna (...:A1NReference) , and optional third and fourth antennas are available.

return no_of_meas: numeric Range: 0 to 30E+3

set_a_0_reference(*no_of_meas: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:NOMeas:LENergy:LE2M:A0Reference
driver.configure.rxQuality.iqDrange.noMeas.lowEnergy.le2M.set_a_0_reference(no_
↪of_meas = 1)
```

Defines the number of measurements per measurement cycle for the specified antenna. Commands for uncoded LE 1M PHY (.. :LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second antenna (...:A1NReference) , and optional third and fourth antennas are available.

param no_of_meas numeric Range: 0 to 30E+3

set_a_1_nreference(*no_of_meas: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:NOMeas:LENergy:LE2M:A1NReference
driver.configure.rxQuality.iqDrange.noMeas.lowEnergy.le2M.set_a_1_nreference(no_
↪of_meas = 1)
```

Defines the number of measurements per measurement cycle for the specified antenna. Commands for uncoded LE 1M PHY (.. :LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second antenna (...:A1NReference) , and optional third and fourth antennas are available.

param no_of_meas numeric Range: 0 to 30E+3

set_a_2_nreference(*no_of_meas: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:NOMeas:LENergy:LE2M:A2NReference
driver.configure.rxQuality.iqDrange.noMeas.lowEnergy.le2M.set_a_2_nreference(no_
↪of_meas = 1)
```

Defines the number of measurements per measurement cycle for the specified antenna. Commands for uncoded LE 1M PHY (.. :LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second antenna (...:A1NReference) , and optional third and fourth antennas are available.

param no_of_meas numeric Range: 0 to 30E+3

set_a_3_nreference(*no_of_meas: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:NOMeas:LENergy:LE2M:A3NReference
driver.configure.rxQuality.iqDrange.noMeas.lowEnergy.le2M.set_a_3_nreference(no_
↪of_meas = 1)
```

Defines the number of measurements per measurement cycle for the specified antenna. Commands for uncoded LE 1M PHY (.. :LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second antenna (...:A1NReference) , and optional third and fourth antennas are available.

param no_of_meas numeric Range: 0 to 30E+3

7.1.12.8.3 Packets

class Packets

Packets commands group definition. 2 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.12.8.3.1 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange:PACKets:LENergy:LE1M
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange:PACKets:LENergy:LE2M
```

class LowEnergy

LowEnergy commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

get_le_1_m() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:PACKets:LENergy:LE1M
value: int = driver.configure.rxQuality.iqDrange.packets.lowEnergy.get_le_1_m()
```

Defines the number of packets to be sent per measurement cycle (statistics cycle) . Commands for uncoded LE 1M PHY (.. :LE1M..) and LE 2M PHY (..:LE2M..) are available.

return packets: numeric Range: 1 to 6.4E+6

get_le_2_m() → int

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:PACKets:LENergy:LE2M
value: int = driver.configure.rxQuality.iqDrange.packets.lowEnergy.get_le_2_m()
```

Defines the number of packets to be sent per measurement cycle (statistics cycle) . Commands for uncoded LE 1M PHY (.. :LE1M..) and LE 2M PHY (..:LE2M..) are available.

return packets: numeric Range: 1 to 6.4E+6

set_le_1_m(*packets: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:PACKets:LENergy:LE1M
driver.configure.rxQuality.iqDrange.packets.lowEnergy.set_le_1_m(packets = 1)
```

Defines the number of packets to be sent per measurement cycle (statistics cycle) . Commands for uncoded LE 1M PHY (.. :LE1M..) and LE 2M PHY (..:LE2M..) are available.

param packets numeric Range: 1 to 6.4E+6

set_le_2_m(*packets: int*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:PACKets:LENergy:LE2M
driver.configure.rxQuality.iqDrange.packets.lowEnergy.set_le_2_m(packets = 1)
```

Defines the number of packets to be sent per measurement cycle (statistics cycle) . Commands for uncoded LE 1M PHY (.. :LE1M..) and LE 2M PHY (..:LE2M..) are available.

param packets numeric Range: 1 to 6.4E+6

7.1.12.8.4 Limit

class Limit

Limit commands group definition. 2 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.12.8.4.1 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange:LIMit:LENergy:LE1M
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange:LIMit:LENergy:LE2M
```

class LowEnergy

LowEnergy commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

class Le1MStruct

Structure for reading output parameters. Fields:

- Ref_Ant_Enable: bool: No parameter help available
- Ant_1_Enable: bool: No parameter help available
- Ant_2_Enable: bool: No parameter help available
- Ant_3_Enable: bool: No parameter help available

class Le2MStruct

Structure for reading output parameters. Fields:

- Ref_Ant_Enable: bool: No parameter help available
- Ant_1_Enable: bool: No parameter help available
- Ant_2_Enable: bool: No parameter help available
- Ant_3_Enable: bool: No parameter help available

get_le_1_m() → Le1MStruct

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:LIMit:LENergy:LE1M
value: Le1MStruct = driver.configure.rxQuality.iqDrange.limit.lowEnergy.get_le_
↪1_m()
```

No command help available

return structure: for return value, see the help for Le1MStruct structure arguments.

get_le_2_m() → Le2MStruct

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:LIMit:LENergy:LE2M
value: Le2MStruct = driver.configure.rxQuality.iqDrange.limit.lowEnergy.get_le_
↪2_m()
```

No command help available

return structure: for return value, see the help for Le2MStruct structure arguments.

set_le_1_m(*value: RsCmwBluetooth-Sig.Implementations.Configure_.RxQuality_.IqDrange_.Limit_.LowEnergy.LowEnergy.Le1MStruct*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:LIMit:LENergy:LE1M
driver.configure.rxQuality.iqDrange.limit.lowEnergy.set_le_1_m(value =␣
↪Le1MStruct())
```

No command help available

param value see the help for Le1MStruct structure arguments.

set_le_2_m(*value: RsCmwBluetooth-Sig.Implementations.Configure_.RxQuality_.IqDrange_.Limit_.LowEnergy.LowEnergy.Le2MStruct*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:LIMit:LENergy:LE2M
driver.configure.rxQuality.iqDrange.limit.lowEnergy.set_le_2_m(value =␣
↪Le2MStruct())
```

No command help available

param value see the help for Le2MStruct structure arguments.

7.1.12.8.5 AntMeanAmp

class AntMeanAmp

AntMeanAmp commands group definition. 2 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.12.8.5.1 Limit

class Limit

Limit commands group definition. 2 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.12.8.5.2 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange:ANTMeanamp:LIMit:LENergy:LE1M
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange:ANTMeanamp:LIMit:LENergy:LE2M
```

class LowEnergy

LowEnergy commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

class Le1MStruct

Structure for reading output parameters. Fields:

- Ant_3_Minus_2_Enable: bool: No parameter help available
- Ant_2_Minus_Ref_Enable: bool: No parameter help available
- Ant_Ref_Minus_1_Enable: bool: No parameter help available

class Le2MStruct

Structure for reading output parameters. Fields:

- Ant_3_Minus_2_Enable: bool: No parameter help available
- Ant_2_Minus_Ref_Enable: bool: No parameter help available
- Ant_Ref_Minus_1_Enable: bool: No parameter help available

get_le_1_m() → Le1MStruct

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:ANTMeanamp:LIMit:LENergy:LE1M
value: Le1MStruct = driver.configure.rxQuality.iqDrange.antMeanAmp.limit.
↪lowEnergy.get_le_1_m()
```

Disables/enables the limit check for the IQ dynamic range results to monitor the requirement: MeanANT3 < MeanANT2 < MeanANT0 < MeanANT1

return structure: for return value, see the help for Le1MStruct structure arguments.

get_le_2_m() → Le2MStruct

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:ANTMeanamp:LIMit:LENergy:LE2M
value: Le2MStruct = driver.configure.rxQuality.iqDrange.antMeanAmp.limit.
↪lowEnergy.get_le_2_m()
```

Disables/enables the limit check for the IQ dynamic range results to monitor the requirement: MeanANT3 < MeanANT2 < MeanANT0 < MeanANT1

return structure: for return value, see the help for Le2MStruct structure arguments.

set_le_1_m(*value: RsCmwBluetoothSig.Implementations.Configure_.RxQuality_.IqDrange_.AntMeanAmp_.Limit_.LowEnergy.LowEnergy.Le1MStruct*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:ANTMeanamp:LIMit:LENergy:LE1M
driver.configure.rxQuality.iqDrange.antMeanAmp.limit.lowEnergy.set_le_1_m(value␣
↪= Le1MStruct())
```

Disables/enables the limit check for the IQ dynamic range results to monitor the requirement: MeanANT3 < MeanANT2 < MeanANT0 < MeanANT1

param value see the help for Le1MStruct structure arguments.

set_le_2_m(*value: RsCmwBluetoothSig.Implementations.Configure_.RxQuality_.IqDrange_.AntMeanAmp_.Limit_.LowEnergy.LowEnergy.Le2MStruct*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:ANTMeanamp:LIMit:LENergy:LE2M
driver.configure.rxQuality.iqDrange.antMeanAmp.limit.lowEnergy.set_le_2_m(value␣
↪= Le2MStruct())
```

Disables/enables the limit check for the IQ dynamic range results to monitor the requirement: MeanANT3 < MeanANT2 < MeanANT0 < MeanANT1

param value see the help for Le2MStruct structure arguments.

7.1.12.9 Itend

class Itend

Itend commands group definition. 2 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.12.9.1 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:ITENd:LENergy:LE1M
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:ITENd:LENergy:LE2M
```

class LowEnergy

LowEnergy commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

get_le_1_m() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:ITENd:LENergy:LE1M
value: bool = driver.configure.rxQuality.itend.lowEnergy.get_le_1_m()
```

Specifies, whether the R&S CMW ignores the DUT´s response to end of direction finding tests.

return ignore_test_end: OFF | ON

get_le_2_m() → bool

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:ITENd:LENergy:LE2M
value: bool = driver.configure.rxQuality.itend.lowEnergy.get_le_2_m()
```

Specifies, whether the R&S CMW ignores the DUT´s response to end of direction finding tests.

return ignore_test_end: OFF | ON

set_le_1_m(*ignore_test_end: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:ITENd:LENergy:LE1M
driver.configure.rxQuality.itend.lowEnergy.set_le_1_m(ignore_test_end = False)
```

Specifies, whether the R&S CMW ignores the DUT´s response to end of direction finding tests.

param ignore_test_end OFF | ON

set_le_2_m(*ignore_test_end: bool*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:ITENd:LENergy:LE2M
driver.configure.rxQuality.itend.lowEnergy.set_le_2_m(ignore_test_end = False)
```

Specifies, whether the R&S CMW ignores the DUT´s response to end of direction finding tests.

param ignore_test_end OFF | ON

7.1.12.10 Cbits

class Cbits

Cbits commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.12.10.1 Tmode

class Tmode
Tmode commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.12.10.1.1 LowEnergy

SCPI Commands

```
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:CBITs:TMODe:LENergy:LE1M
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:CBITs:TMODe:LENergy:LE2M
CONFigure:BLUetooth:SIGNaling<Instance>:RXQuality:CBITs:TMODe:LENergy:LRANge
```

class LowEnergy
LowEnergy commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class Le1MStruct
Structure for reading output parameters. Fields:

- No_Bits_To_Corrupt: int: No parameter help available
- Byte_Start_Err: int: No parameter help available

class Le2MStruct
Structure for reading output parameters. Fields:

- No_Bits_To_Corrupt: int: No parameter help available
- Byte_Start_Err: int: No parameter help available

class LrangeStruct
Structure for reading output parameters. Fields:

- No_Bits_To_Corrupt: int: No parameter help available
- Byte_Start_Err: int: No parameter help available

get_le_1_m() → Le1MStruct

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:CBITs:TMODe:LENergy:LE1M
value: Le1MStruct = driver.configure.rxQuality.cbits.tmode.lowEnergy.get_le_1_
↪m()
```

No command help available

return structure: for return value, see the help for Le1MStruct structure arguments.

get_le_2_m() → Le2MStruct

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:CBITs:TMODe:LENergy:LE2M
value: Le2MStruct = driver.configure.rxQuality.cbits.tmode.lowEnergy.get_le_2_
↪m()
```

No command help available

return structure: for return value, see the help for Le2MStruct structure arguments.

get_lrange() → LrangeStruct

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:CBITs:TMODe:LENergy:LRANge
value: LrangeStruct = driver.configure.rxQuality.cbits.tmode.lowEnergy.get_
↪lrange()
```

No command help available

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

set_le_1_m(*value: RsCmwBluetooth-Sig.Implementations.Configure_.RxQuality_.Cbits_.Tmode_.LowEnergy.LowEnergy.Le1MStruct*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:CBITs:TMODe:LENergy:LE1M
driver.configure.rxQuality.cbits.tmode.lowEnergy.set_le_1_m(value =␣
↪Le1MStruct())
```

No command help available

param value see the help for Le1MStruct structure arguments.

set_le_2_m(*value: RsCmwBluetooth-Sig.Implementations.Configure_.RxQuality_.Cbits_.Tmode_.LowEnergy.LowEnergy.Le2MStruct*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:CBITs:TMODe:LENergy:LE2M
driver.configure.rxQuality.cbits.tmode.lowEnergy.set_le_2_m(value =␣
↪Le2MStruct())
```

No command help available

param value see the help for Le2MStruct structure arguments.

set_lrange(*value: RsCmwBluetooth-Sig.Implementations.Configure_.RxQuality_.Cbits_.Tmode_.LowEnergy.LowEnergy.LrangeStruct*) → None

```
# SCPI: CONFigure:BLUetooth:SIGNaling<Instance>
↪:RXQuality:CBITs:TMODe:LENergy:LRANge
driver.configure.rxQuality.cbits.tmode.lowEnergy.set_lrange(value =␣
↪LrangeStruct())
```

No command help available

param value see the help for LrangeStruct structure arguments.

7.2 Diagnostic

SCPI Commands

```
DIAGnostic:BLUetooth:SIGNaling<Instance>:WCMap
```

class Diagnostic

Diagnostic commands group definition. 22 total commands, 6 Sub-groups, 1 group commands

get_wcmap() → int

```
# SCPI: DIAGnostic:BLUetooth:SIGNaling<Instance>:WCMap
value: int = driver.diagnostic.get_wcmap()
```

No command help available

return wait_con_intervals: No help available

set_wcmap(*wait_con_intervals: int*) → None

```
# SCPI: DIAGnostic:BLUetooth:SIGNaling<Instance>:WCMap
driver.diagnostic.set_wcmap(wait_con_intervals = 1)
```

No command help available

param wait_con_intervals No help available

Cloning the Group

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

Subgroups

7.2.1 Delay

SCPI Commands

```
DIAGnostic:BLUetooth:SIGNaling<Instance>:DELay:PTIMeout
DIAGnostic:BLUetooth:SIGNaling<Instance>:DELay:TMODe
```

class Delay

Delay commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

get_ptimeout() → int

```
# SCPI: DIAGnostic:BLUetooth:SIGNaling<Instance>:DELay:PTIMeout
value: int = driver.diagnostic.delay.get_ptimeout()
```

No command help available

return poll_timeout: No help available

get_tmode() → int

```
# SCPI: DIAGnostic:BLUetooth:SIGNaling<Instance>:DELay:TMODe
value: int = driver.diagnostic.delay.get_tmode()
```

No command help available

return act_test_delay: No help available

set_ptimeout(*poll_timeout: int*) → None

```
# SCPI: DIAGnostic:BLUetooth:SIGNaling<Instance>:DELay:PTIMeout
driver.diagnostic.delay.set_ptimeout(poll_timeout = 1)
```

No command help available

param poll_timeout No help available

set_tmode(*act_test_delay: int*) → None

```
# SCPI: DIAGnostic:BLUetooth:SIGNaling<Instance>:DELay:TMODe
driver.diagnostic.delay.set_tmode(act_test_delay = 1)
```

No command help available

param act_test_delay No help available

7.2.2 Le

SCPI Commands

```
DIAGnostic:BLUetooth:SIGNaling<Instance>:LE:MODE
DIAGnostic:BLUetooth:SIGNaling<Instance>:LE:STATe
DIAGnostic:BLUetooth:SIGNaling<Instance>:LE:PLENgth
DIAGnostic:BLUetooth:SIGNaling<Instance>:LE:CHANnel
DIAGnostic:BLUetooth:SIGNaling<Instance>:LE:PATTern
```

class Le

Le commands group definition. 5 total commands, 0 Sub-groups, 5 group commands

get_channel() → int

```
# SCPI: DIAGnostic:BLUetooth:SIGNaling<Instance>:LE:CHANnel
value: int = driver.diagnostic.le.get_channel()
```

No command help available

return channel: No help available

get_mode() → bool

```
# SCPI: DIAGnostic:BLUetooth:SIGNaling<Instance>:LE:MODE
value: bool = driver.diagnostic.le.get_mode()
```

No command help available

return le_test_mode: No help available

get_packet_length() → int

```
# SCPI: DIAGnostic:BLUetooth:SIGNaling<Instance>:LE:PLENgth
value: int = driver.diagnostic.le.get_packet_length()
```

No command help available

return payload_length: No help available

get_pattern() → RsCmwBluetoothSig.enums.LeRangePaternType

```
# SCPI: DIAGnostic:BLUetooth:SIGNaling<Instance>:LE:PATTern
value: enums.LeRangePaternType = driver.diagnostic.le.get_pattern()
```

No command help available

return pattern: No help available

get_state() → RsCmwBluetoothSig.enums.LeDiagState

```
# SCPI: DIAGnostic:BLUetooth:SIGNaling<Instance>:LE:STATe
value: enums.LeDiagState = driver.diagnostic.le.get_state()
```

No command help available

return state: No help available

set_channel(*channel: int*) → None

```
# SCPI: DIAGnostic:BLUetooth:SIGNaling<Instance>:LE:CHANnel
driver.diagnostic.le.set_channel(channel = 1)
```

No command help available

param channel No help available

set_mode(*le_test_mode: bool*) → None

```
# SCPI: DIAGnostic:BLUetooth:SIGNaling<Instance>:LE:MODE
driver.diagnostic.le.set_mode(le_test_mode = False)
```

No command help available

param le_test_mode No help available

set_packet_length(*payload_length: int*) → None

```
# SCPI: DIAGnostic:BLUetooth:SIGNaling<Instance>:LE:PLENgth
driver.diagnostic.le.set_packet_length(payload_length = 1)
```

No command help available

param payload_length No help available

set_pattern(*pattern: RsCmwBluetoothSig.enums.LeRangePaternType*) → None

```
# SCPI: DIAGnostic:BLUetooth:SIGNaling<Instance>:LE:PATTern
driver.diagnostic.le.set_pattern(pattern = enums.LeRangePaternType.ALL0)
```

No command help available

param pattern No help available

7.2.3 Ucs

SCPI Commands

```
DIAGnostic:BLUetooth:SIGNaling<Instance>:UCS:STATe
DIAGnostic:BLUetooth:SIGNaling<Instance>:UCS:FREQuency
DIAGnostic:BLUetooth:SIGNaling<Instance>:UCS:MODE
DIAGnostic:BLUetooth:SIGNaling<Instance>:UCS:TESTvector
```

class Ucs

Ucs commands group definition. 4 total commands, 0 Sub-groups, 4 group commands

class FrequencyStruct

Structure for reading output parameters. Fields:

- Cmw_Rx_Frequency: float: No parameter help available
- Cmw_Tx_Frequency: float: No parameter help available

get_frequency() → FrequencyStruct

```
# SCPI: DIAGnostic:BLUetooth:SIGNaling<Instance>:UCS:FREQuency
value: FrequencyStruct = driver.diagnostic.ucs.get_frequency()
```

No command help available

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

get_mode() → bool

```
# SCPI: DIAGnostic:BLUetooth:SIGNaling<Instance>:UCS:MODE
value: bool = driver.diagnostic.ucs.get_mode()
```

No command help available

return ucs_mode: No help available

get_state() → RsCmwBluetoothSig.enums.LeDiagState

```
# SCPI: DIAGnostic:BLUetooth:SIGNaling<Instance>:UCS:STATe
value: enums.LeDiagState = driver.diagnostic.ucs.get_state()
```

No command help available

return state: No help available

get_test_vector() → RsCmwBluetoothSig.enums.TestVector

```
# SCPI: DIAGnostic:BLUetooth:SIGNaling<Instance>:UCS:TESTvector
value: enums.TestVector = driver.diagnostic.ucs.get_test_vector()
```

No command help available

return test_vector: No help available

set_frequency(*value: RsCmwBluetoothSig.Implementations.Diagnostic_.Ucs.Ucs.FrequencyStruct*) → None

```
# SCPI: DIAGnostic:BLUetooth:SIGNaling<Instance>:UCS:FREQuency
driver.diagnostic.ucs.set_frequency(value = FrequencyStruct())
```

No command help available

param value see the help for FrequencyStruct structure arguments.

set_mode(*ucs_mode: bool*) → None

```
# SCPI: DIAGnostic:BLUetooth:SIGNaling<Instance>:UCS:MODE
driver.diagnostic.ucs.set_mode(ucs_mode = False)
```

No command help available

param ucs_mode No help available

set_test_vector(*test_vector: RsCmwBluetoothSig.enums.TestVector*) → None

```
# SCPI: DIAGnostic:BLUetooth:SIGNaling<Instance>:UCS:TESTvector
driver.diagnostic.ucs.set_test_vector(test_vector = enums.TestVector.INITstack)
```

No command help available

param test_vector No help available

7.2.4 Debug

SCPI Commands

```
DIAGnostic:BLUetooth:SIGNaling<Instance>:DEBug:SUALog
DIAGnostic:BLUetooth:SIGNaling<Instance>:DEBug:SUAFswlog
DIAGnostic:BLUetooth:SIGNaling<Instance>:DEBug:SIMulation
DIAGnostic:BLUetooth:SIGNaling<Instance>:DEBug:HCIWindow
DIAGnostic:BLUetooth:SIGNaling<Instance>:DEBug:APPWindow
DIAGnostic:BLUetooth:SIGNaling<Instance>:DEBug:ATTRibwindow
```

class Debug

Debug commands group definition. 8 total commands, 1 Sub-groups, 6 group commands

get_app_window() → bool

```
# SCPI: DIAGnostic:BLUetooth:SIGNaling<Instance>:DEBug:APPWindow
value: bool = driver.diagnostic.debug.get_app_window()
```

No command help available

return window: No help available

get_attr_window() → bool

```
# SCPI: DIAGnostic:BLUetooth:SIGNaling<Instance>:DEBug:ATTRibwindow
value: bool = driver.diagnostic.debug.get_attr_window()
```

No command help available

return window: No help available

get_hci_window() → bool

```
# SCPI: DIAGnostic:BLUetooth:SIGNaling<Instance>:DEBug:HCIWindow
value: bool = driver.diagnostic.debug.get_hci_window()
```

No command help available

return window: No help available

get_simulation() → bool

```
# SCPI: DIAGnostic:BLUetooth:SIGNaling<Instance>:DEBug:SIMulation
value: bool = driver.diagnostic.debug.get_simulation()
```

No command help available

return simulation: No help available

get_sua_fsw_log() → bool

```
# SCPI: DIAGnostic:BLUetooth:SIGNaling<Instance>:DEBug:SUAFswlog
value: bool = driver.diagnostic.debug.get_sua_fsw_log()
```

No command help available

return sua_fsw_log: No help available

get_sua_log() → bool

```
# SCPI: DIAGnostic:BLUetooth:SIGNaling<Instance>:DEBug:SUALog
value: bool = driver.diagnostic.debug.get_sua_log()
```

No command help available

return sua_log_win: No help available

set_app_window(*window: bool*) → None

```
# SCPI: DIAGnostic:BLUetooth:SIGNaling<Instance>:DEBug:APPWindow
driver.diagnostic.debug.set_app_window(window = False)
```

No command help available

param window No help available

set_attr_window(*window: bool*) → None

```
# SCPI: DIAGnostic:BLUetooth:SIGNaling<Instance>:DEBug:ATTRibwindow
driver.diagnostic.debug.set_attr_window(window = False)
```

No command help available

param window No help available

set_hci_window(*window: bool*) → None

```
# SCPI: DIAGnostic:BLUetooth:SIGNaling<Instance>:DEBug:HCIWindow
driver.diagnostic.debug.set_hci_window(window = False)
```

No command help available

param window No help available

set_simulation(*simulation: bool*) → None

```
# SCPI: DIAGnostic:BLUetooth:SIGNaling<Instance>:DEBug:SIMulation
driver.diagnostic.debug.set_simulation(simulation = False)
```

No command help available

param simulation No help available

set_sua_fsw_log(*sua_fsw_log: bool*) → None

```
# SCPI: DIAGnostic:BLUetooth:SIGNaling<Instance>:DEBug:SUAFswlog
driver.diagnostic.debug.set_sua_fsw_log(sua_fsw_log = False)
```

No command help available

param sua_fsw_log No help available

set_sua_log(*sua_log_win: bool*) → None

```
# SCPI: DIAGnostic:BLUetooth:SIGNaling<Instance>:DEBug:SUALog
driver.diagnostic.debug.set_sua_log(sua_log_win = False)
```

No command help available

param sua_log_win No help available

Cloning the Group

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

Subgroups

7.2.4.1 LinkLayer

SCPI Commands

```
DIAGnostic:BLUetooth:SIGNaling<Instance>:DEBug:LINKlayer:IPADdress
DIAGnostic:BLUetooth:SIGNaling<Instance>:DEBug:LINKlayer:PORTaddress
```

class LinkLayer

LinkLayer commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

get_ip_address() → str

```
# SCPI: DIAGnostic:BLUetooth:SIGNaling<Instance>:DEBug:LINKlayer:IPADdress
value: str = driver.diagnostic.debug.linkLayer.get_ip_address()
```

No command help available

return window: No help available

get_port_address() → str

```
# SCPI: DIAGnostic:BLUetooth:SIGNaling<Instance>:DEBug:LINKlayer:PORTaddress
value: str = driver.diagnostic.debug.linkLayer.get_port_address()
```

No command help available

return window: No help available

set_ip_address(*window: str*) → None

```
# SCPI: DIAGnostic:BLUetooth:SIGNaling<Instance>:DEBug:LINKlayer:IPADdress
driver.diagnostic.debug.linkLayer.set_ip_address(window = '1')
```

No command help available

param window No help available

set_port_address(*window: str*) → None

```
# SCPI: DIAGnostic:BLUetooth:SIGNaling<Instance>:DEBug:LINKlayer:PORTaddress
driver.diagnostic.debug.linkLayer.set_port_address(window = '1')
```

No command help available

param window No help available

7.2.5 Connection

class Connection

Connection commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.2.5.1 Packets

class Packets

Packets commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.2.5.1.1 EpLength

class EpLength
EpLength commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.diagnostic.connection.packets.epLength.clone()
```

Subgroups

7.2.5.1.1.1 LowEnergy

SCPI Commands

```
DIAGnostic:BLUetooth:SIGNaling<Instance>:CONNection:PACKets:EPLength:LENergy:LE2M
```

class LowEnergy
LowEnergy commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_le_2_m() → int

```
# SCPI: DIAGnostic:BLUetooth:SIGNaling<Instance>
↪:CONNection:PACKets:EPLength:LENergy:LE2M
value: int = driver.diagnostic.connection.packets.epLength.lowEnergy.get_le_2_
↪m()
```

No command help available

return payload_length: No help available

set_le_2_m(*payload_length: int*) → None

```
# SCPI: DIAGnostic:BLUetooth:SIGNaling<Instance>
↪:CONNection:PACKets:EPLength:LENergy:LE2M
driver.diagnostic.connection.packets.epLength.lowEnergy.set_le_2_m(payload_
↪length = 1)
```

No command help available

param payload_length No help available

7.2.6 RxQuality

SCPI Commands

```
DIAGnostic:BLUetooth:SIGNaling<Instance>:RXQuality:PERShow
```

class RxQuality

RxQuality commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

set_per_show(*per_show: bool*) → None

```
# SCPI: DIAGnostic:BLUetooth:SIGNaling<Instance>:RXQuality:PERShow
driver.diagnostic.rxQuality.set_per_show(per_show = False)
```

No command help available

param per_show No help available

7.3 Sense

SCPI Commands

```
SENSe:BLUetooth:SIGNaling<Instance>:CMAP
```

class Sense

Sense commands group definition. 35 total commands, 4 Sub-groups, 1 group commands

get_cmap() → List[int]

```
# SCPI: SENSe:BLUetooth:SIGNaling<Instance>:CMAP
value: List[int] = driver.sense.get_cmap()
```

Queries channels used by adaptive frequency hopping (AFH) .

return afh_channel_map: decimal 79 comma-separated values, one value per channel: 0: channel is blocked for AFH 1: channel is released for AFH Range: 0 to 1

Cloning the Group

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

Subgroups

7.3.1 UsbDevice

class UsbDevice
UsbDevice commands group definition. 8 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.3.1.1 Information

SCPI Commands

```
SENSe:BLUetooth:SIGNaling<Instance>:USBDevice:INFormation:DPRotocol
SENSe:BLUetooth:SIGNaling<Instance>:USBDevice:INFormation:DSUBclass
SENSe:BLUetooth:SIGNaling<Instance>:USBDevice:INFormation:DCLass
SENSe:BLUetooth:SIGNaling<Instance>:USBDevice:INFormation:IDPRoduct
SENSe:BLUetooth:SIGNaling<Instance>:USBDevice:INFormation:IDVendor
SENSe:BLUetooth:SIGNaling<Instance>:USBDevice:INFormation:PRODuct
SENSe:BLUetooth:SIGNaling<Instance>:USBDevice:INFormation:SERial
SENSe:BLUetooth:SIGNaling<Instance>:USBDevice:INFormation:MANufacturer
```

class Information
Information commands group definition. 8 total commands, 0 Sub-groups, 8 group commands

get_dclass() → str

```
# SCPI: SENSe:BLUetooth:SIGNaling<Instance>:USBDevice:INFormation:DCLass
value: str = driver.sense.usbDevice.information.get_dclass()
```

Returns the device class of the active USB device.

return name: string

get_dprotocol() → str

```
# SCPI: SENSe:BLUetooth:SIGNaling<Instance>:USBDevice:INFormation:DPRotocol
value: str = driver.sense.usbDevice.information.get_dprotocol()
```

Returns the supported protocol of the active USB device.

return name: string

get_dsub_class() → str

```
# SCPI: SENSe:BLUetooth:SIGNaling<Instance>:USBDevice:INFormation:DSUBclass
value: str = driver.sense.usbDevice.information.get_dsub_class()
```

Returns the device subclass of the active USB device.

return name: string

get_id_vendor() → str

```
# SCPI: SENSe:BLUetooth:SIGNaling<Instance>:USBDevice:INFormation:IDVendor
value: str = driver.sense.usbDevice.information.get_id_vendor()
```

Returns the vendor ID of the active USB device.

return name: string

get_idproduct() → str

```
# SCPI: SENSe:BLUetooth:SIGNaling<Instance>:USBDevice:INFormation:IDPRoduct
value: str = driver.sense.usbDevice.information.get_idproduct()
```

Returns the product ID of the active USB device.

return name: string

get_manufacturer() → str

```
# SCPI: SENSe:BLUetooth:SIGNaling<Instance>:USBDevice:INFormation:MANufacturer
value: str = driver.sense.usbDevice.information.get_manufacturer()
```

Returns the name of the manufacturer of the active USB device.

return name: string

get_product() → str

```
# SCPI: SENSe:BLUetooth:SIGNaling<Instance>:USBDevice:INFormation:PRODuct
value: str = driver.sense.usbDevice.information.get_product()
```

Returns the product name of the active USB device.

return name: string

get_serial() → str

```
# SCPI: SENSe:BLUetooth:SIGNaling<Instance>:USBDevice:INFormation:SERial
value: str = driver.sense.usbDevice.information.get_serial()
```

Returns the name serial of the active USB device.

return name: string

7.3.2 Eut

class Eut
Eut commands group definition. 23 total commands, 4 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.3.2.1 Capability

SCPI Commands

```
SENSe:BLUetooth:SIGNaling<Instance>:EUT:CAPability:ESCO
SENSe:BLUetooth:SIGNaling<Instance>:EUT:CAPability:SCLass
SENSe:BLUetooth:SIGNaling<Instance>:EUT:CAPability:ENCRyption
SENSe:BLUetooth:SIGNaling<Instance>:EUT:CAPability:PCONtrol
SENSe:BLUetooth:SIGNaling<Instance>:EUT:CAPability:EPControl
SENSe:BLUetooth:SIGNaling<Instance>:EUT:CAPability:PSAVing
SENSe:BLUetooth:SIGNaling<Instance>:EUT:CAPability:CONNection
SENSe:BLUetooth:SIGNaling<Instance>:EUT:CAPability:SCO
SENSe:BLUetooth:SIGNaling<Instance>:EUT:CAPability:ACL
SENSe:BLUetooth:SIGNaling<Instance>:EUT:CAPability:LESignaling
```

class Capability
Capability commands group definition. 11 total commands, 1 Sub-groups, 10 group commands

class AclStruct
Structure for reading output parameters. Fields:

- Fc_Lag: int: decimal Flow control lag Range: 0 to 7 , Unit: 256 bytes
- Dh_3_Dm_3: bool: OFF | ON Three-slot packets
- Dh_5_Dm_5: bool: OFF | ON Five-slot packets
- Edr_3_Slot: bool: OFF | ON Three-slot EDR ACL packets
- Edr_5_Slot: bool: OFF | ON Five-slot EDR ACL packets
- Edr_2_Mbps: bool: OFF | ON EDR ACL 2 Mbit/s
- Edr_3_Mbps: bool: OFF | ON EDR ACL 3 Mbit/s

class ConnectionStruct
Structure for reading output parameters. Fields:

- Page_Scan_Mode: enums.PageScanMode: 0X00 | 0X01 | 0X02 | 0X03 The page scan mode that is used for default page scan: 0X00: mandatory page scan mode 0X01: optional page scan mode I 0X02: optional page scan mode II 0X03: optional page scan mode III
- Pg_Scan_Prd_Mode: enums.PageScanPeriodMode: P0 | P1 | P2 Page scan period mode
- Pg_Scan_Rep_Mode: enums.PsrMode: R0 | R1 | R2 Page scan repetition mode

- Pscheme: bool: OFF | ON ‘Optional paging scheme’ support
- Slot_Offset: bool: OFF | ON ‘Slot offset’ support
- Timing_Acc: bool: OFF | ON ‘Timing accuracy’ support
- Switch: bool: OFF | ON ‘Switching between master and slave’ support
- Rssi: bool: OFF | ON ‘Received signal strength indication’ support

class EscoStruct

Structure for reading output parameters. Fields:

- Ev_3_Packets: bool: OFF | ON EV3 packets
- Ev_4_Packets: bool: OFF | ON EV4 packets
- Ev_5_Packets: bool: OFF | ON EV5 packets
- Three_Slot_Edr_Pck: bool: OFF | ON 2-EV5 / 3-EV5 packets over three slots
- Edr_2_Mbps_Mode: bool: OFF | ON 2-EV3 / 2-EV5 packets
- Edr_3_Mbps_Mode: bool: OFF | ON 3-EV3 / 3-EV5 packets

class LeSignalingStruct

Structure for reading output parameters. Fields:

- Device_Name: str: string LE device name
- Le_Encryption: bool: OFF | ON Support of LE encryption
- Conn_Param_Req: bool: OFF | ON Support of connection parameter request procedure
- Ext_Rejection_Ind: bool: OFF | ON Support of extended rejection indication
- Slave_Init_Feat_Ex: bool: OFF | ON Support of slave-initiated features exchange
- Le_Ping: bool: OFF | ON Support of LE ping
- Le_Data_Pk_Lex: bool: No parameter help available
- Ll_Privacy: bool: OFF | ON Support of LL privacy
- Ex_Scan_Filt_Pol: bool: OFF | ON Support of extended scanner filter policies
- Le_2_Mphy: bool: OFF | ON Support of LE 2 Mbps PHY
- Stab_Mod_Trans: bool: OFF | ON Support of stable modulation index for transmitter
- Stab_Mod_Rec: bool: OFF | ON Support of stable modulation index for receiver
- Le_Coded_Phy: bool: No parameter help available
- Le_Ext_Adv: bool: OFF | ON Support of extended advertising
- Le_Periodic_Adv: bool: No parameter help available
- Chan_Sel_Alg_2: bool: OFF | ON Support of channel selection algorithm No. 2
- Le_Power_Class_1: bool: No parameter help available
- Min_Number_Used_Ch: bool: No parameter help available

class PsavingStruct

Structure for reading output parameters. Fields:

- Hold: bool: OFF | ON Hold mode
- Sniff: bool: OFF | ON Sniff mode

- Park: bool: OFF | ON Park mode

class ScoStruct

Structure for reading output parameters. Fields:

- Cqdd: bool: OFF | ON Channel quality driven data rate
- Link: bool: OFF | ON SCO link
- Ts_Data: bool: OFF | ON Transparent SCO data
- Hv_2_P: bool: OFF | ON HV2 packets
- Hv_3_P: bool: OFF | ON HV3 packets
- Ulaw: bool: OFF | ON -law log
- Alaw: bool: OFF | ON A-law log
- Cvsd: bool: OFF | ON Continuous variable slope delta modulation

get_acl() → AclStruct

```
# SCPI: SENSe:BLUetooth:SIGNaling<Instance>:EUT:CAPability:ACL
value: AclStruct = driver.sense.eut.capability.get_acl()
```

Gets the ACL-related capabilities of the connected EUT. Except for the flow control lag, for each capability either OFF or ON is returned.

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

get_connection() → ConnectionStruct

```
# SCPI: SENSe:BLUetooth:SIGNaling<Instance>:EUT:CAPability:CONNection
value: ConnectionStruct = driver.sense.eut.capability.get_connection()
```

Gets the connection-related properties and capabilities of the connected EUT.

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

get_encryption() → bool

```
# SCPI: SENSe:BLUetooth:SIGNaling<Instance>:EUT:CAPability:ENCRyption
value: bool = driver.sense.eut.capability.get_encryption()
```

Queries whether the connected EUT supports encryption (OFF|ON)

return encryption: OFF | ON

get_ep_control() → bool

```
# SCPI: SENSe:BLUetooth:SIGNaling<Instance>:EUT:CAPability:EPControl
value: bool = driver.sense.eut.capability.get_ep_control()
```

Queries whether the connected EUT supports enhanced power control (OFF|ON)

return epower_control: OFF | ON

get_esco() → EscoStruct

```
# SCPI: SENSe:BLUetooth:SIGNaling<Instance>:EUT:CAPability:ESCO
value: EscoStruct = driver.sense.eut.capability.get_esco()
```

Gets the e-SCO-related capabilities of the connected EUT. For each capability, either OFF or ON is returned

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

get_le_signaling() → LeSignalingStruct

```
# SCPI: SENSe:BLUetooth:SIGNaling<Instance>:EUT:CAPability:LESignaling
value: LeSignalingStruct = driver.sense.eut.capability.get_le_signaling()
```

Queries capabilities retrieved from the EUT.

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

get_power_control() → bool

```
# SCPI: SENSe:BLUetooth:SIGNaling<Instance>:EUT:CAPability:PCONtrol
value: bool = driver.sense.eut.capability.get_power_control()
```

Queries whether the connected EUT supports legacy power control (OFF|ON)

return power_control: OFF | ON

get_psaving() → PsavingStruct

```
# SCPI: SENSe:BLUetooth:SIGNaling<Instance>:EUT:CAPability:PSAVing
value: PsavingStruct = driver.sense.eut.capability.get_psaving()
```

Gets the power saving-related capabilities of the connected EUT. For each capability, either OFF or ON is returned

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

get_sclass() → str

```
# SCPI: SENSe:BLUetooth:SIGNaling<Instance>:EUT:CAPability:SCLass
value: str = driver.sense.eut.capability.get_sclass()
```

Gets the (major) service class of the connected EUT as specified in https://www.bluetooth.org/Technical/AssignedNumbers/baseband.htm.

return service_class: string

get_sco() → ScoStruct

```
# SCPI: SENSe:BLUetooth:SIGNaling<Instance>:EUT:CAPability:SCO
value: ScoStruct = driver.sense.eut.capability.get_sco()
```

Gets the SCO-related capabilities of the connected EUT. For each capability, either OFF or ON is returned

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

Cloning the Group

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

Subgroups

7.3.2.1.1 Adp

SCPI Commands

```
SENSe:BLUetooth:SIGNaling<Instance>:EUT:CAPability:ADP:SBC
```

class Adp

Adp commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class SbcStruct

Structure for reading output parameters. Fields:

- S_16_K: bool: No parameter help available
- S_32_K: bool: No parameter help available
- S_44_K: bool: No parameter help available
- S_48_K: bool: No parameter help available
- Cmmn: bool: No parameter help available
- Cmdl: bool: No parameter help available
- Cmst: bool: No parameter help available
- Cmjs: bool: No parameter help available
- B_04_B: bool: No parameter help available
- B_08_B: bool: No parameter help available
- B_12_B: bool: No parameter help available
- B_16_B: bool: No parameter help available
- Sb_4_B: bool: No parameter help available
- Sb_8_B: bool: No parameter help available
- Allocation: enums.AllocMethod: No parameter help available
- Min_Bit_Pool: int: No parameter help available
- Max_Bit_Pool: int: No parameter help available

get_sbc() → SbcStruct

```
# SCPI: SENSe:BLUetooth:SIGNaling<Instance>:EUT:CAPability:ADP[:SBC]
value: SbcStruct = driver.sense.eut.capability.adp.get_sbc()
```

No command help available

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

7.3.2.2 Information

SCPI Commands

```
SENSe:BLUetooth:SIGNaling<Instance>:EUT:INFormation:BDADdress
SENSe:BLUetooth:SIGNaling<Instance>:EUT:INFormation:CLASs
SENSe:BLUetooth:SIGNaling<Instance>:EUT:INFormation:COMPany
SENSe:BLUetooth:SIGNaling<Instance>:EUT:INFormation:NAME
SENSe:BLUetooth:SIGNaling<Instance>:EUT:INFormation:VERSion
SENSe:BLUetooth:SIGNaling<Instance>:EUT:INFormation:LESignaling
```

class Information

Information commands group definition. 6 total commands, 0 Sub-groups, 6 group commands

class ClassStruct

Structure for reading output parameters. Fields:

- Major_Dev_Class: str: string
- Minor_Dev_Class: str: string

class LeSignalingStruct

Structure for reading output parameters. Fields:

- Device_Name: str: string
- Address_Type: enums.AddressTypeExt: PUBLic | RANDom | PIDentity | RSIDentity Public device address, random device address: random private ID, random static ID
- Gm_Discoverable: bool: OFF | ON GAP mode general discoverable
- Gm_Limit_Disc: bool: OFF | ON GAP mode limited discoverable
- Gm_Connectable: bool: OFF | ON GAP mode undirected connectable
- Gm_Direct_Con: bool: OFF | ON GAP mode directed connectable

class VersionStruct

Structure for reading output parameters. Fields:

- Lmp: str: string Textual representation of the LMP version (VersNr) as defined in http://www.bluetooth.org/Technical/AssignedNumbers/link_manager.htm
- Company: str: string Textual representation of the company identifier (CompId) as defined in http://www.bluetooth.org/Technical/AssignedNumbers/identifiers.htm
- Lmp_Subversion: str: string LMP subversion number (SubVersNr) , a company-internal version number

get_bd_address() → str

```
# SCPI: SENSe:BLUetooth:SIGNaling<Instance>:EUT:INFormation:BDADdress
value: str = driver.sense.eut.information.get_bd_address()
```

Gets the address (BD_ADDR) of the connected Bluetooth device.

return bd_address: hex The Bluetooth device address in hexadecimal notation. Range: #H0 to #HFFFFFFFFFFFF (12 hexadecimal digits)

get_class() → ClassStruct

```
# SCPI: SENSe:BLUetooth:SIGNaling<Instance>:EUT:INFormation:CLASs
value: ClassStruct = driver.sense.eut.information.get_class()
```

Gets the major and the minor device class of the connected EUT as specified in https://www.bluetooth.org/Technical/AssignedNumbers/baseband.htm.

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

get_company() → str

```
# SCPI: SENSe:BLUetooth:SIGNaling<Instance>:EUT:INFormation:COMPany
value: str = driver.sense.eut.information.get_company()
```

Queries the company identifier (CompId) as defined in http://www.bluetooth.org/Technical/AssignedNumbers/identifiers.htm

return company: string

get_le_signaling() → LeSignalingStruct

```
# SCPI: SENSe:BLUetooth:SIGNaling<Instance>:EUT:INFormation:LESignaling
value: LeSignalingStruct = driver.sense.eut.information.get_le_signaling()
```

Retrieves EUT information as device name, device address and GAP mode.

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

get_name() → str

```
# SCPI: SENSe:BLUetooth:SIGNaling<Instance>:EUT:INFormation:NAME
value: str = driver.sense.eut.information.get_name()
```

Gets the name of the connected device.

return name: string Up to 255 characters

get_version() → VersionStruct

```
# SCPI: SENSe:BLUetooth:SIGNaling<Instance>:EUT:INFormation:VERSion
value: VersionStruct = driver.sense.eut.information.get_version()
```

Gets LMP_version_res information.

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

7.3.2.3 Csettings

SCPI Commands

```
SENSe:BLUetooth:SIGNaling<Instance>:EUT:CSETtings:LESignaling
```

class Csettings

Csettings commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class LeSignalingStruct

Structure for reading output parameters. Fields:

- Connection_Int: float: float Range: 6 to 3200
- Supervision_Tout: float: float Supervision timeout Range: 100 ms to 32E+3 ms
- Slave_Latency: float: float Slave latency Range: 0 to 499

get_le_signaling() → LeSignalingStruct

```
# SCPI: SENSe:BLUetooth:SIGNaling<Instance>:EUT:CSETtings:LESignaling
value: LeSignalingStruct = driver.sense.eut.csettings.get_le_signaling()
```

Queries the connection parameters for LE normal mode.

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

7.3.2.4 PowerControl

class PowerControl

PowerControl commands group definition. 5 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.3.2.4.1 State

SCPI Commands

```
SENSe:BLUetooth:SIGNaling<Instance>:EUT:PCONtrol:STATe:LESignaling
SENSe:BLUetooth:SIGNaling<Instance>:EUT:PCONtrol:STATe:GFSK
SENSe:BLUetooth:SIGNaling<Instance>:EUT:PCONtrol:STATe:DQPSk
SENSe:BLUetooth:SIGNaling<Instance>:EUT:PCONtrol:STATe:DPSK
SENSe:BLUetooth:SIGNaling<Instance>:EUT:PCONtrol:STATe
```

class State

State commands group definition. 5 total commands, 0 Sub-groups, 5 group commands

class DpskStruct
Structure for reading output parameters. Fields:

- Power_Change: enums.PowerChange: NNE | UP | DOWN | MAX NNE: none UP: power up command accepted DOWN: power down command accepted MAX: maximum power command accepted
- Power_Min_Max: enums.PowerMinMax: NOTS | CHANged | MAX | MIN | NNM NOTS: not supported (command not accepted by the EUT) CHANged: changed one step MAX: max power reached MIN: min power reached NNM: no new message

class DqpskStruct
Structure for reading output parameters. Fields:

- Power_Change: enums.PowerChange: NNE | UP | DOWN | MAX NNE: none UP: power up command accepted DOWN: power down command accepted MAX: maximum power command accepted
- Power_Min_Max: enums.PowerMinMax: NOTS | CHANged | MAX | MIN | NNM NOTS: not supported (command not accepted by the EUT) CHANged: changed one step MAX: max power reached MIN: min power reached NNM: no new message

class GfskStruct
Structure for reading output parameters. Fields:

- Power_Change: enums.PowerChange: NNE | UP | DOWN | MAX NNE: none UP: power up command accepted DOWN: power down command accepted MAX: maximum power command accepted
- Power_Min_Max: enums.PowerMinMax: NOTS | CHANged | MAX | MIN | NNM NOTS: not supported (command not accepted by the EUT) CHANged: changed one step MAX: max power reached MIN: min power reached NNM: no new message

class LeSignalingStruct
Structure for reading output parameters. Fields:

- Pow_Flag: enums.PowerFlag: NONE | MIN | MAX NONE: no new message MIN: minimum power command accepted MAX: maximum power command accepted
- Pow_Delta: int: decimal Returns the power difference before and after the command execution Range: -100 dB to +100 dB
- Current_Power: int: decimal Returns the actual Tx power level of the DUT Range: -200 dBm to +100 dBm
- Eut_State: enums.EutState: FAIL | OK Checks the command execution.

class ValueStruct
Structure for reading output parameters. Fields:

- Pow_Change_Gfsk: enums.PowerChange: NNE | UP | DOWN | MAX NNE: none UP: power up command accepted DOWN: power down command accepted MAX: maximum power command accepted
- Pow_Min_Max_Gfsk: enums.PowerMinMax: NOTS | CHANged | MAX | MIN | NNM NOTS: not supported (command not accepted by the EUT) CHANged: changed one step MAX: max power reached MIN: min power reached NNM: no new message
- Pow_Change_Dqpsk: enums.PowerChange: NNE | UP | DOWN | MAX
- Pow_Min_Max_Dqpsk: enums.PowerMinMax: NOTS | CHANged | MAX | MIN | NNM
- Pow_Change_Dpsk: enums.PowerChange: NNE | UP | DOWN | MAX
- Pow_Min_Max_Dpsk: enums.PowerMinMax: NOTS | CHANged | MAX | MIN | NNM

get_dpsk() → DpskStruct

```
# SCPI: SENSe:BLUetooth:SIGNaling<Instance>:EUT:PCONtrol:STATe:DPSK
value: DpskStruct = driver.sense.eut.powerControl.state.get_dpsk()
```

Displays the EUT responses to the enhanced power control state commands for GFSK-modulated BR and /4 DQPSK or 8DPSK-modulated EDR packets. The modulation is indicated by the last mnemonic.

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

get_dqpsk() → DqpskStruct

```
# SCPI: SENSe:BLUetooth:SIGNaling<Instance>:EUT:PCONtrol:STATe:DQPSk
value: DqpskStruct = driver.sense.eut.powerControl.state.get_dqpsk()
```

Displays the EUT responses to the enhanced power control state commands for GFSK-modulated BR and /4 DQPSK or 8DPSK-modulated EDR packets. The modulation is indicated by the last mnemonic.

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

get_gfsk() → GfskStruct

```
# SCPI: SENSe:BLUetooth:SIGNaling<Instance>:EUT:PCONtrol:STATe:GFSK
value: GfskStruct = driver.sense.eut.powerControl.state.get_gfsk()
```

Displays the EUT responses to the enhanced power control state commands for GFSK-modulated BR and /4 DQPSK or 8DPSK-modulated EDR packets. The modulation is indicated by the last mnemonic.

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

get_le_signaling() → LeSignalingStruct

```
# SCPI: SENSe:BLUetooth:SIGNaling<Instance>:EUT:PCONtrol:STATe:LESignaling
value: LeSignalingStruct = driver.sense.eut.powerControl.state.get_le_
↪signaling()
```

Displays the DUT's responses to the power control state commands for low energy.

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

get_value() → ValueStruct

```
# SCPI: SENSe:BLUetooth:SIGNaling<Instance>:EUT:PCONtrol:STATe
value: ValueStruct = driver.sense.eut.powerControl.state.get_value()
```

Displays the EUT responses to the enhanced power control state commands for GFSK-modulated BR and /4 DQPSK and 8DPSK-modulated EDR packets.

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

7.3.3 Connection

class Connection
Connection commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.3.3.1 Audio

SCPI Commands

```
SENSe:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:LINFo
```

class Audio
Audio commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class LinfoStruct
Structure for reading output parameters. Fields:

- Speech_Code: enums.SpeechCode: CVSD | ALAW | ULAW | MSBC CVSD (8 kHz) , A-law (8 kHz) , -law (8 kHz) , or mSBC (16 kHz) codec
- Link_Type: enums.VoiceLinkType: SCO | ESCO Synchronous connection-oriented (SCO) or enhanced synchronous connection-oriented (eSCO) link
- Packet_Type: enums.PacketTypeEsco: HV1 | HV2 | HV3 | EV3 | EV4 | EV5 | 2EV3 | 3EV3 | 2EV5 | 3EV5 HV1: SCO packets high-quality voice, 1/3 rate FEC HV2: SCO packets high-quality voice, 2/3 rate FEC HV3: SCO packets high-quality voice, no FEC EV3, EV4, EV5: eSCO packets on the top of BR ACL connection 2EV3, 3EV3, 2EV5, 3EV5: eSCO packets on the top of EDR ACL connection (2-EV3, 3-EV3, 2-EV5, 3-EV5 packets)
- Sample_Rate: float: float Range: 0 kHz to 999 kHz
- Data_Rate: int: decimal Range: 0 kbit/s to 9999 kbit/s

get_linfo() → LinfoStruct

```
# SCPI: SENSe:BLUetooth:SIGNaling<Instance>:CONNection:AUDio:LINFo
value: LinfoStruct = driver.sense.connection.audio.get_linfo()
```

Queries the parameters of active audio connection.

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

7.3.4 Elogging

SCPI Commands

```
SENSe:BLUetooth:SIGNaling<Instance>:ELOGging:LAST
SENSe:BLUetooth:SIGNaling<Instance>:ELOGging:ALL
```

class Elogging

Elogging commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

class AllStruct

Structure for reading output parameters. Fields:

- Timestamp: List[str]: string Timestamp of the entry as string in the format 'hh:mm:ss'
- Category: List[enums.LogCategory]: INFO | WARNing | ERRor | CONTinue Category of the entry, as indicated in the main view by an icon CONTinue means the continuation of previous entry.
- Event: List[str]: string Text string describing the event

class LastStruct

Structure for reading output parameters. Fields:

- Timestamp: str: string Timestamp of the entry as string in the format 'hh:mm:ss'
- Category: enums.LogCategory: INFO | WARNing | ERRor | CONTinue Category of the entry, as indicated in the main view by an icon CONTinue means the continuation of previous entry.
- Event: str: string Text string describing the event

get_all() → AllStruct

```
# SCPI: SENSe:BLUetooth:SIGNaling<Instance>:ELOGging:ALL
value: AllStruct = driver.sense.elogging.get_all()
```

Queries all entries of the event log. For each entry three parameters are returned, from oldest to latest entry: {<Timestamp>, <Category>, <Event>}entry 1, {<Timestamp>, <Category>, <Event>}entry 2, ...

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

get_last() → LastStruct

```
# SCPI: SENSe:BLUetooth:SIGNaling<Instance>:ELOGging:LAST
value: LastStruct = driver.sense.elogging.get_last()
```

Queries the latest entry of the event log.

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

7.4 Connection

class Connection
Connection commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.4.1 State

SCPI Commands

```
FETCh:BLUetooth:SIGNaling<Instance>:CONNection:STATe
```

class State
State commands group definition. 3 total commands, 2 Sub-groups, 1 group commands

fetch() → RsCmwBluetoothSig.enums.ConnectionState

```
# SCPI: FETCh:BLUetooth:SIGNaling<Instance>:CONNection:STATe
value: enums.ConnectionState = driver.connection.state.fetch()
```

Returns the signaling state of the R&S CMW for BR/EDR. State changes are initiated using the method RsCmwBluetoothSig. Call.Connection.Action.value command.

return state: OFF | SBY | INQuiring | SINQuiry | CNNecting | SCONnecting | CONNected | DETaching | TCNNecting | TCONected | ECRunning | ECNNecting | ECONected | EXEMode | ENEMode | HFCNnecting | HFConnected | EXHFp | ENHFp | AGCNnecting | AGConnected | EXAGmode | ENAGmode | ENHSmode | EXHSmode | CNASmode | CHASmode | DHASmode | EHASmode | XHASmode | SMIDle | SMCNnecting | SMConnected | SMDetaching | HSCNnecting | HSConnected | HSDetaching | A2CNnecting | A2Connected | A2Detaching | A2SNnecting | A2SCnnected | A2SDetaching | ACNNecting | ACONected | AEXMode | AENMode OFF: not connected SBY: standby INQuiring: inquiring SINQuiry: stopping inquiry CNNecting: connecting SCONnecting: stop connecting CONNected: connected DETaching: detaching TCNNecting: test mode - connecting TCONnected: test mode - connected ECRunning: EUT controller running ECNNecting: audio echo mode - connecting ECONected: audio echo mode - connected EXEMode: audio echo mode - exiting ENEMode: audio echo mode - entering HFCNnecting: hands-free profile - connecting HFConnected: hands-free profile - connected EXHFp: hands-free profile - exiting ENHFp: hands-free profile - entering AGCNnecting: hands-free audio gateway profile - connecting AGConnected: hands-free audio gateway profile - connected EXAGmode: hands-free audio gateway profile - exiting ENAGmode: hands-free audio gateway profile - entering CNASmode: hands-free audio gateway (slave mode) - connecting CHASmode: hands-free audio gateway (slave mode) - connected DHASmode: hands-free audio gateway (slave mode) - detaching EHASmode: hands-free

audio gateway (slave mode) - entering XHASmode: hands-free audio gateway (slave mode) - exiting SMIDle: slave mode - idle SMCNnecting: slave mode - connecting SMConnected: slave mode - connected SMDetaching: slave mode - detaching HSCNnecting: hands-free profile (slave mode) - connecting HSConnected: hands-free profile (slave mode) - connected ENHSmode: hands-free profile (slave mode) - entering EXHSmode: hands-free profile (slave mode) - exiting HSDetaching: hands-free profile (slave mode) - detaching A2CNnecting: A2DP - connecting A2Connected: A2DP - connected A2Detaching: A2DP - detaching A2SNnecting: A2DP (slave mode) - connecting A2SCnnected: A2DP (slave mode) - connected A2SDetaching: A2DP (slave mode) - detaching ACNNecting: audio - connecting ACONected: audio - connected AEXMode: audio - exiting AENMode: audio - entering For a detailed description of the available states and state transitions, see 'Signaling States'.

Cloning the Group

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

Subgroups

7.4.1.1 All

SCPI Commands

```
FETCh:BLUetooth:SIGNaling<Instance>:CONNection:STATe:ALL
```

class All

All commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Main_State: bool: OFF | ON OFF: Generator switched off ON: Generator has been turned on
- Sub_State: enums.ConnectionState: OFF | SBY | INQuiring | SINQuiry | CNNecting | SCONnecting | CONNected | DETaching | TCNNecting | TCONected | ECRunning | ECNNecting | ECONected | EXEMode | ENEMode | HFCNnecting | HFConnected | EXHFp | ENHFp | AGCNnecting | AGConnected | EXAGmode | ENAGmode | ENHSmode | EXHSmode | CNASmode | CHASmode | DHASmode | EHASmode | XHASmode | SMIDle | SMCNnecting | SMConnected | SMDetaching | HSCNnecting | HSConnected | HSDetaching | A2CNnecting | A2Connected | A2Detaching | A2SNnecting | A2SCnnected | A2SDetaching | ACNNecting | ACONected | AEXMode | AENMode Signaling states of the R&S CMW for BR/EDR: OFF: not connected SBY: standby INQuiring: inquiring SINQuiry: stopping inquiry CNNecting: connecting SCONnecting: stop connecting CONNected: connected DETaching: detaching TCNNecting: test mode - connecting TCONnected: test mode - connected ECRunning: EUT controller running ECNNecting: audio echo mode - connecting ECONected: audio echo mode - connected EXEMode: audio echo mode - exiting ENEMode: audio echo mode - entering HFCNnecting: hands-free profile - connecting HFConnected: hands-free profile - connected EXHFp: hands-free profile - exiting ENHFp: hands-free profile - entering AGCNnecting: hands-free audio gateway profile - connecting AGConnected: hands-free audio gateway profile - connected EXAGmode: hands-free audio gateway profile - exiting ENAGmode: hands-free audio gateway profile - entering CNASmode: hands-free audio gateway (slave mode) - connecting CHASmode: hands-free audio gateway (slave mode) - connected DHASmode: hands-free audio gateway (slave mode) - detaching EHASmode: hands-free audio gateway (slave mode) - entering XHASmode: hands-free audio

gateway (slave mode) - exiting SMIDle: slave mode - idle SMCNnecting: slave mode - connecting SMConnected: slave mode - connected SMDetaching: slave mode - detaching HSCNnecting: hands-free profile (slave mode) - connecting HSConnected: hands-free profile (slave mode) - connected ENHSmode: hands-free profile (slave mode) - entering EXHSmode: hands-free profile (slave mode) - exiting HSDetaching: hands-free profile (slave mode) - detaching A2CNnecting: A2DP - connecting A2Connected: A2DP - connected A2Detaching: A2DP - detaching A2SNnecting: A2DP (slave mode) - connecting A2SCnnected: A2DP (slave mode) - connected A2SDetaching: A2DP (slave mode) - detaching ACNNecting: audio - connecting ACONected: audio - connected AEXMode: audio - exiting AENMode: audio - entering For a detailed description of the available states and state transitions, see 'Signaling States'.

fetch() → FetchStruct

```
# SCPI: FETCh:BLUetooth:SIGNaling<Instance>:CONNection:STATe:ALL
value: FetchStruct = driver.connection.state.all.fetch()
```

Returns detailed information about the 'Bluetooth Signaling' generator state.

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

7.4.1.2 LeSignaling

SCPI Commands

```
FETCh:BLUetooth:SIGNaling<Instance>:CONNection:STATe:LESignaling
```

class LeSignaling

LeSignaling commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch() → RsCmwBluetoothSig.enums.SignalingState

```
# SCPI: FETCh:BLUetooth:SIGNaling<Instance>:CONNection:STATe:LESignaling
value: enums.SignalingState = driver.connection.state.leSignaling.fetch()
```

Returns the signaling state of the R&S CMW for LE. State changes are initiated using the method RsCmwBluetoothSig.Call. Connection.Action.leSignaling command.

return state: OFF | SBY | INQuiring | CNNecting | CONNected | DETaching | TCONected | TCNNecting | CPOWer OFF: not connected SBY: standby INQuiring: inquiring CNNecting: connecting CONNected: connected DETaching: detaching TCONected: LE test mode - connected CPOWer: LE power control

7.5 Source

SCPI Commands

```
SOURce:BLUetooth:SIGNaling<Instance>:STATe
```

class Source

Source commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_state() → bool

```
# SCPI: SOURce:BLUetooth:SIGNaling<Instance>:STATe
value: bool = driver.source.get_state()
```

Sets/gets the main state of the 'Bluetooth Signaling' application. Signaling actions such as inquiry or paging are initiated using the method RsCmwBluetoothSig.Call.Connection.Action.value command.

return main_state: ON | OFF | 1 | 0 When turned ON, the R&S CMW switches to standby state (see method RsCmwBluetoothSig.Connection.State.fetch)

set_state(*main_state: bool*) → None

```
# SCPI: SOURce:BLUetooth:SIGNaling<Instance>:STATe
driver.source.set_state(main_state = False)
```

Sets/gets the main state of the 'Bluetooth Signaling' application. Signaling actions such as inquiry or paging are initiated using the method RsCmwBluetoothSig.Call.Connection.Action.value command.

param main_state ON | OFF | 1 | 0 When turned ON, the R&S CMW switches to standby state (see method RsCmwBluetoothSig.Connection.State.fetch)

7.6 Route

class Route

Route commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.6.1 Scenario

SCPI Commands

```
ROUTe:BLUetooth:SIGNaling<Instance>:SCENario:STATe
```

class Scenario

Scenario commands group definition. 3 total commands, 1 Sub-groups, 1 group commands

get_state() → RsCmwBluetoothSig.enums.ConnectionState

```
# SCPI: ROUTe:BLUetooth:SIGNaling<Instance>:SCENario:STATe
value: enums.ConnectionState = driver.route.scenario.get_state()
```

No command help available

return state: No help available

Cloning the Group

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

Subgroups

7.6.1.1 OtRx

SCPI Commands

```
ROUTe:BLUetooth:SIGNaling<Instance>:SCENario:OTRX:FLEXible
ROUTe:BLUetooth:SIGNaling<Instance>:SCENario:OTRX
```

class OtRx

OtRx commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

class FlexibleStruct

Structure for reading output parameters. Fields:

- Bb_Board: enums.BbBoard: Signaling unit
- Rx_Connector: enums.RxConnector: RF connector for the input path
- Rx_Converter: enums.RxConverter: RX module for the input path
- Tx_Connector: enums.TxConnector: RF connector for the output path
- Tx_Converter: enums.TxConverter: TX module for the output path

class ValueStruct

Structure for reading output parameters. Fields:

- Rx_Connector: enums.RxConnector: RF connector for the input path
- Rx_Converter: enums.RxConverter: RX module for the input path
- Tx_Connector: enums.TxConnector: RF connector for the output path
- Tx_Converter: enums.TxConverter: TX module for the output path

get_flexible() → FlexibleStruct

```
# SCPI: ROUTe:BLUetooth:SIGNaling<Instance>:SCENario:OTRX:FLEXible
value: FlexibleStruct = driver.route.scenario.otRx.get_flexible()
```

Activates the scenario with one output and one input signal paths. For possible signaling unit, connector and converter values, see 'Values for Signal Path Selection'.

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

get_value() → ValueStruct

```
# SCPI: ROUTe:BLUetooth:SIGNaling<Instance>:SCENario:OTRX
value: ValueStruct = driver.route.scenario.otRx.get_value()
```

Activates the scenario with one output and one input signal paths. For possible connector and converter values, see 'Values for Signal Path Selection'.

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

set_flexible(*value: RsCmwBluetoothSig.Implementations.Route_.Scenario_.OtRx.OtRx.FlexibleStruct*) → None

```
# SCPI: ROUTe:BLUetooth:SIGNaling<Instance>:SCENario:OTRX:FLEXible
driver.route.scenario.otRx.set_flexible(value = FlexibleStruct())
```

Activates the scenario with one output and one input signal paths. For possible signaling unit, connector and converter values, see 'Values for Signal Path Selection'.

param value see the help for FlexibleStruct structure arguments.

set_value(*value: RsCmwBluetoothSig.Implementations.Route_.Scenario_.OtRx.OtRx.ValueStruct*) → None

```
# SCPI: ROUTe:BLUetooth:SIGNaling<Instance>:SCENario:OTRX
driver.route.scenario.otRx.set_value(value = ValueStruct())
```

Activates the scenario with one output and one input signal paths. For possible connector and converter values, see 'Values for Signal Path Selection'.

param value see the help for ValueStruct structure arguments.

7.7 Call

class Call

Call commands group definition. 9 total commands, 5 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.7.1 HciCustom

SCPI Commands

```
CALL:BLUetooth:SIGNaling<Instance>:HCICustom:SEND
```

class HciCustom

HciCustom commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

send(*custom_hci_byte: Optional[List[str]] = None*) → None

```
# SCPI: CALL:BLUetooth:SIGNaling<Instance>:HCICustom:SEND
driver.call.hciCustom.send(custom_hci_byte = ['raw1', 'raw2', 'raw3'])
```

Sends specified bytes in hexadecimal format as an HCI command via USB interface. You can send multiple bytes separated by comma.

param custom_hci_byte hex Comma-separated hexadecimal string Range: #H0 to #HFF

7.7.2 Rdevices

SCPI Commands

```
CALL:BLUetooth:SIGNaling<Instance>:RDEVices
```

class Rdevices

Rdevices commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

set() → None

```
# SCPI: CALL:BLUetooth:SIGNaling<Instance>:RDEVices
driver.call.rdevices.set()
```

Detects the DUTs connected to USB ports directly or via a USB-to-COM adapter.

set_with_opc() → None

```
# SCPI: CALL:BLUetooth:SIGNaling<Instance>:RDEVices
driver.call.rdevices.set_with_opc()
```

Detects the DUTs connected to USB ports directly or via a USB-to-COM adapter.

Same as set, but waits for the operation to complete before continuing further. Use the RsCmwBluetoothSig.utilities.opc_timeout_set() to set the timeout value.

7.7.3 DtMode

class DtMode

DtMode commands group definition. 2 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.7.3.1 EndTx

SCPI Commands

```
CALL:BLUetooth:SIGNaling:DTMode:ENDTx
```

class EndTx

EndTx commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

set() → None

```
# SCPI: CALL:BLUetooth:SIGNaling:DTMode:ENDTx
driver.call.dtMode.endTx.set()
```

No command help available

set_with_opc() → None

```
# SCPI: CALL:BLUetooth:SIGNaling:DTMode:ENDTx
driver.call.dtMode.endTx.set_with_opc()
```

No command help available

Same as set, but waits for the operation to complete before continuing further. Use the RsCmwBluetoothSig.utilities.opc_timeout_set() to set the timeout value.

7.7.3.2 StartTx

SCPI Commands

```
CALL:BLUetooth:SIGNaling:DTMode:STARttx
```

class StartTx

StartTx commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

set() → None

```
# SCPI: CALL:BLUetooth:SIGNaling:DTMode:STARttx
driver.call.dtMode.startTx.set()
```

No command help available

set_with_opc() → None

```
# SCPI: CALL:BLUetooth:SIGNaling:DTMode:STARttx
driver.call.dtMode.startTx.set_with_opc()
```

No command help available

Same as set, but waits for the operation to complete before continuing further. Use the RsCmwBluetoothSig.utilities.opc_timeout_set() to set the timeout value.

7.7.4 LowEnergy

SCPI Commands

```
CALL:BLUetooth:SIGNaling<Instance>:LENergy:RESet
```

class LowEnergy

LowEnergy commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

reset() → None

```
# SCPI: CALL:BLUetooth:SIGNaling<Instance>:LENergy:RESet
driver.call.lowEnergy.reset()
```

Sends the HCI reset command to the EUT via USB.

reset_with_opc() → None

```
# SCPI: CALL:BLUetooth:SIGNaling<Instance>:LENergy:RESet
driver.call.lowEnergy.reset_with_opc()
```

Sends the HCI reset command to the EUT via USB.

Same as reset, but waits for the operation to complete before continuing further. Use the RsCmwBluetoothSig.utilities.opc_timeout_set() to set the timeout value.

7.7.5 Connection

SCPI Commands

```
CALL:BLUetooth:SIGNaling<Instance>:CONNection:ACONnect
```

class Connection

Connection commands group definition. 4 total commands, 2 Sub-groups, 1 group commands

get_aconnect() → bool

```
# SCPI: CALL:BLUetooth:SIGNaling<instance>:CONNection:ACONnect
value: bool = driver.call.connection.get_aconnect()
```

No command help available

return auto_ranging: No help available

set_aconnect(*auto_ranging: bool*) → None

```
# SCPI: CALL:BLUetooth:SIGNaling<instance>:CONNection:ACONnect
driver.call.connection.set_aconnect(auto_ranging = False)
```

No command help available

param auto_ranging No help available

Cloning the Group

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

Subgroups

7.7.5.1 Check

SCPI Commands

```
CALL:BLUetooth:SIGNaling<Instance>:CONNection:CHECk:LENergy
```

class Check

Check commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_low_energy() → RsCmwBluetoothSig.enums.ConTestResult

```
# SCPI: CALL:BLUetooth:SIGNaling<Instance>:CONNection:CHECk:LENergy
value: enums.ConTestResult = driver.call.connection.check.get_low_energy()
```

Checks the LE connection via USB.

return con_test_result: PASS | FAIL | TOUT | NRUN Result: passed, failed, timeout, not running

7.7.5.2 Action

SCPI Commands

```
CALL:BLUetooth:SIGNaling<Instance>:CONNection:ACTion:LESignaling
CALL:BLUetooth:SIGNaling<Instance>:CONNection:ACTion
```

class Action

Action commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

set_le_signaling(*action: RsCmwBluetoothSig.enums.ConnectionActionLe*) → None

```
# SCPI: CALL:BLUetooth:SIGNaling<Instance>:CONNection:ACTion:LESignaling
driver.call.connection.action.set_le_signaling(action = enums.
↪ConnectionActionLe.CONNect)
```

Requests the R&S CMW to perform certain signaling actions for LE. It has no query form: the current signaling state can be retrieved using the method RsCmwBluetoothSig.Connection.State.LeSignaling.fetch command.

param action INQuire | SINQuiry | CONNect | SCONnecting | DETach | TMConnect INQuire: Switch on master signal and start inquiry for Bluetooth devices within range Inquiry stops after a configurable maximum duration (see method RsCmwBluetoothSig.Configure.Connection.Inquiry.Duration.leSignaling) or after a configurable number of responses (see method RsCmwBluetoothSig.Configure.Connection.Inquiry.NoResponses.leSignaling) SINQuiry: Stop inquiry, switch off master signal and return to standby state CONNect: Switch on master signal, start paging the selected Bluetooth device and establish an ACL connection SCONnecting: Stop an ongoing connection setup, switch off the master signal and return to standby state DETach: Detach an established connection, switch off the master signal and return to standby state TMConnect: Connect test mode - switch on master signal, start paging the selected Bluetooth device, establish a LE test mode connection, and transmit test packets

set_value(*action: RsCmwBluetoothSig.enums.ConnectAction*) → None

```
# SCPI: CALL:BLUetooth:SIGNaling<Instance>:CONNection:ACTion
driver.call.connection.action.set_value(action = enums.ConnectAction.ADConnect)
```

Requests the R&S CMW to perform certain signaling actions for BR/EDR. It has no query form: the current signaling state can be retrieved using the method RsCmwBluetoothSig.Connection.State.fetch command.

param action INQuire | SINQuiry | SCONnecting | STMode | CONNect | TMConnect | DETach | REController | EMConnect | EXEMode | ENEMode | HFPConnect | EXHFp | ENHFp | AGConnect | ENAGate | EXAGate | ADConnect | AUDConnect | ADEXit | ADENter INQuire: Switch on master signal and start inquiry for Bluetooth devices within range Inquiry stops after a configurable maximum duration (see method RsCmwBluetoothSig.Configure.Connection.Inquiry.ilength) or after a configurable number of responses (see method RsCmwBluetoothSig.Configure.Connection.Inquiry.NoResponses.value) SINQuiry: Stop inquiry, switch off master signal and return to standby state SCONnecting: Stop an ongoing connection setup, switch off the master signal and return to standby state STMode: Stop a test mode connection, switch off the master signal and return to standby state CONNect: Switch on master signal, start paging the selected Bluetooth device and establish an ACL connection TMConnect: Switch on master signal, start paging the selected Bluetooth device and establish a test mode connection DETach: Detach an established connection, switch off the master signal and return to standby state REController: Run EUT controller to reset and initialize the EUT via USB connection EMConnect: Connect audio echo mode EXEMode: Exit audio echo mode ENEMode: Enter audio echo mode HFPConnect: Connect hands-free profile EXHFp: Exit hands-free profile ENHFp: Enter hands-free profile AGConnect: Connect hands-free audio gateway profile ENAGate: Enter hands-free audio gateway profile EXAGate: Exit hands-free audio gateway profile ADConnect: Connect A2DP AUDConnect: Connect audio mode ADEXit: Exit audio mode ADENter: Enter audio mode

7.8 LowEnergy

class LowEnergy
LowEnergy commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.8.1 State

SCPI Commands

```
FETCh:BLUetooth:SIGNaling<Instance>:LENergy:STATe
```

class State
State commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch() → RsCmwBluetoothSig.enums.LeSignalingState

```
# SCPI: FETCh:BLUetooth:SIGNaling<Instance>:LENergy:STATe
value: enums.LeSignalingState = driver.lowEnergy.state.fetch()
```

Returns the signaling state of the R&S CMW for LE direct test connections.

return state: IDLE | OFF | SPCM | STCM | CMR | STTX | TXRunning | SPTX | STRX | RXRunning | SPRX | RCOM IDLE: Connected in direct test mode, no active test running OFF: Not connected in direct test mode SPCM: Stopping communication test STCM: Starting communication test CMR: Communication test running STTX: Starting TX test TXRunning: TX test running SPTX: Stopping TX test STRX: Starting RX test RXRunning: RX test running SPRX: Stopping RX test RCOM: Refreshing COM port list

7.9 RxQuality

class RxQuality
RxQuality commands group definition. 150 total commands, 6 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.9.1 IqDrange

SCPI Commands

```
INITiate:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange
STOP:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange
ABORt:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange
```

class IqDrange

IqDrange commands group definition. 35 total commands, 3 Sub-groups, 4 group commands

abort() → None

```
# SCPI: ABORt:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange
driver.rxQuality.iqDrange.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.

abort_with_opc() → None

```
# SCPI: ABORt:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange
driver.rxQuality.iqDrange.abort_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.
```

(continues on next page)

(continued from previous page)

```
    - 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 abort, but waits for the operation to complete before continuing further. Use the RsCmwBluetoothSig.utilities.opc_timeout_set() to set the timeout value.

fetch() → RsCmwBluetoothSig.enums.ResourceState

```
# SCPI: FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange
value: enums.ResourceState = driver.rxQuality.iqDrange.fetch()
```

No command help available

return meas_status: No help available

initiate() → None

```
# SCPI: INITiate:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange
driver.rxQuality.iqDrange.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.

initiate_with_opc() → None

```
# SCPI: INITiate:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange
driver.rxQuality.iqDrange.initiate_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.
```

(continues on next page)

(continued from previous page)

Use FETCh...STATe? to query the current measurement state.

Same as initiate, but waits for the operation to complete before continuing further. Use the RsCmwBluetoothSig.utilities.opc_timeout_set() to set the timeout value.

stop() → None

```
# SCPI: STOP:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange
driver.rxQuality.iqDrange.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() → None

```
# SCPI: STOP:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange
driver.rxQuality.iqDrange.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 RsCmwBluetoothSig.utilities.opc_timeout_set() to set the timeout value.

Cloning the Group

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

Subgroups

7.9.1.1 State

SCPI Commands

```
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange:STATe
```

class State

State commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch() → RsCmwBluetoothSig.enums.ResourceState

```
# SCPI: FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange:STATe
value: enums.ResourceState = driver.rxQuality.iqDrange.state.fetch()
```

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.

return meas_status: OFF | RDY | RUN OFF: measurement switched off, no resources allocated, no results available (when entered after ABORt...) RDY: measurement has been terminated, valid results can be available RUN: measurement running (after INITiate..., READ...) , synchronization pending or adjusted, resources active or queued

7.9.1.2 LowEnergy

class LowEnergy

LowEnergy commands group definition. 24 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.9.1.2.1 Le1M

class Le1M

Le1M commands group definition. 12 total commands, 4 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.rxQuality.iqDrange.lowEnergy.le1M.clone()
```

Subgroups

7.9.1.2.1.1 A0Reference

SCPI Commands

```
READ:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange:LENergy:LE1M:A0Reference
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange:LENergy:LE1M:A0Reference
CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange:LENergy:LE1M:A0Reference
```

class A0Reference

A0Reference commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Trans_Packets: float: decimal No. of transmitted packets Range: 0 to 200E+6
- Rec_Events: float: No parameter help available
- Rec_Events_Val_Crc: float: No parameter help available
- No_Valid_Iqp_Airs: enums.ResultStatus2: decimal No. of valid measurements Range: 0 to 100E+3
- Valid_Iq_Pairs_Rec: enums.ResultStatus2: float Percentage of received valid IQ sample pairs related to all received IQ pairs Range: 0 % to 100 % , Unit: %
- Mean_Amplitude: enums.ResultStatus2: float Mean amplitude Range: 0 to 1 , Unit: dBFS
- Max_Amplitude: enums.ResultStatus2: float Max amplitude Range: 0 to 1 , Unit: dBFS
- Rec_Events_Cnt: enums.ResultStatus2: decimal No. of received events
- Pro_Events_Cnt: enums.ResultStatus2: decimal No. of processed events
- Pro_Events_Val_Crc: enums.ResultStatus2: No parameter help available
- Nos_Pos_Amp: enums.ResultStatus2: No parameter help available

class ResultData

Response structure. Fields:

- Trans_Packets: int: No parameter help available
- Rec_Events: float: No parameter help available
- Rec_Events_Val_Crc: int: No parameter help available
- No_Valid_Iqp_Airs: int: No parameter help available
- Valid_Iq_Pairs_Rec: float: No parameter help available
- Mean_Amplitude: float: No parameter help available
- Max_Amplitude: float: No parameter help available
- Rec_Events_Cnt: int: No parameter help available

- Pro_Events_Cnt: int: No parameter help available
- Pro_Events_Val_Crc: int: No parameter help available
- Nof_Pos_Amp: int: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:LENergy:LE1M:A0Reference
value: CalculateStruct = driver.rxQuality.iqDrange.lowEnergy.le1M.a0Reference.
↪calculate()
```

Returns the results of IQ dynamic range Rx measurement for the specified antenna. Commands for uncoded LE 1M PHY (..:LE1M. .) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second non-reference antenna (...:A1NReference) , and optional third and fourth non-reference antennas are available. The 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 RsCmwBluetoothSig.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:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:LENergy:LE1M:A0Reference
value: ResultData = driver.rxQuality.iqDrange.lowEnergy.le1M.a0Reference.fetch()
```

Returns the results of IQ dynamic range Rx measurement for the specified antenna. Commands for uncoded LE 1M PHY (..:LE1M. .) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second non-reference antenna (...:A1NReference) , and optional third and fourth non-reference antennas are available. The 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 RsCmwBluetoothSig.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:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:LENergy:LE1M:A0Reference
value: ResultData = driver.rxQuality.iqDrange.lowEnergy.le1M.a0Reference.read()
```

No command help available

Use RsCmwBluetoothSig.reliability.last_value to read the updated reliability indicator.

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

7.9.1.2.1.2 A1Nreference

SCPI Commands

```
READ:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange:LENergy:LE1M:A1NReference
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange:LENergy:LE1M:A1NReference
CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange:LENergy:LE1M:A1NReference
```

class A1Nreference

A1Nreference commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Trans_Packets: float: decimal No. of transmitted packets Range: 0 to 200E+6
- Rec_Events: float: No parameter help available
- Rec_Events_Val_Crc: float: No parameter help available
- No_Valid_Iqp_Airs: enums.ResultStatus2: decimal No. of valid measurements Range: 0 to 100E+3
- Valid_Iq_Pairs_Rec: enums.ResultStatus2: float Percentage of received valid IQ sample pairs related to all received IQ pairs Range: 0 % to 100 % , Unit: %
- Mean_Amplitude: enums.ResultStatus2: float Mean amplitude Range: 0 to 1 , Unit: dBFS
- Max_Amplitude: enums.ResultStatus2: float Max amplitude Range: 0 to 1 , Unit: dBFS
- Rec_Events_Cnt: enums.ResultStatus2: decimal No. of received events
- Pro_Events_Cnt: enums.ResultStatus2: decimal No. of processed events
- Pro_Events_Val_Crc: enums.ResultStatus2: No parameter help available
- Nos_Pos_Amp: enums.ResultStatus2: No parameter help available

class FetchStruct

Response structure. Fields:

- Trans_Packets: int: decimal No. of transmitted packets Range: 0 to 200E+6
- Rec_Events: float: No parameter help available
- Rec_Events_Val_Crc: int: No parameter help available
- No_Valid_Iqp_Airs: int: decimal No. of valid measurements Range: 0 to 100E+3
- Valid_Iq_Pairs_Rec: float: float Percentage of received valid IQ sample pairs related to all received IQ pairs Range: 0 % to 100 % , Unit: %
- Mean_Amplitude: float: float Mean amplitude Range: 0 to 1 , Unit: dBFS
- Max_Amplitude: float: float Max amplitude Range: 0 to 1 , Unit: dBFS
- Rec_Events_Cnt: int: decimal No. of received events
- Pro_Events_Cnt: int: decimal No. of processed events
- Pro_Events_Val_Crc: int: No parameter help available
- Nof_Pos_Amp: int: decimal No. of possible amplitude measurements determined by CTE length, slot length and number of antennas

class ReadStruct

Response structure. Fields:

- Trans_Packets: float: decimal No. of transmitted packets Range: 0 to 200E+6
- Rec_Events: float: No parameter help available
- Rec_Events_Val_Crc: float: No parameter help available
- No_Valid_Iqp_Airs: float: decimal No. of valid measurements Range: 0 to 100E+3
- Valid_Iq_Pairs_Rec: float: float Percentage of received valid IQ sample pairs related to all received IQ pairs Range: 0 % to 100 % , Unit: %
- Mean_Amplitude: float: float Mean amplitude Range: 0 to 1 , Unit: dBFS
- Max_Amplitude: float: float Max amplitude Range: 0 to 1 , Unit: dBFS
- Rec_Events_Cnt: int: decimal No. of received events
- Pro_Events_Cnt: int: decimal No. of processed events
- Pro_Events_Val_Crc: int: No parameter help available
- Nof_Pos_Amp: int: decimal No. of possible amplitude measurements determined by CTE length, slot length and number of antennas

calculate() → CalculateStruct

```
# SCPI: CALCulate:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:LENergy:LE1M:A1NReference
value: CalculateStruct = driver.rxQuality.iqDrange.lowEnergy.le1M.a1Nreference.
↪calculate()
```

Returns the results of IQ dynamic range Rx measurement for the specified antenna. Commands for uncoded LE 1M PHY (..:LE1M. .) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second non-reference antenna (...:A1NReference) , and optional third and fourth non-reference antennas are available. The 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 RsCmwBluetoothSig.reliability.last_value to read the updated reliability indicator.

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

fetch() → FetchStruct

```
# SCPI: FETCh:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:LENergy:LE1M:A1NReference
value: FetchStruct = driver.rxQuality.iqDrange.lowEnergy.le1M.a1Nreference.
↪fetch()
```

Returns the results of IQ dynamic range Rx measurement for the specified antenna. Commands for uncoded LE 1M PHY (..:LE1M. .) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second non-reference antenna (...:A1NReference) , and optional third and fourth non-reference antennas are available. The 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 RsCmwBluetoothSig.reliability.last_value to read the updated reliability indicator.

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

read() → ReadStruct

```
# SCPI: READ:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:LENergy:LE1M:A1NReference
value: ReadStruct = driver.rxQuality.iqDrange.lowEnergy.le1M.a1Nreference.read()
```

Returns the results of IQ dynamic range Rx measurement for the specified antenna. Commands for uncoded LE 1M PHY (..:LE1M. .) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second non-reference antenna (...:A1NReference) , and optional third and fourth non-reference antennas are available. The 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 RsCmwBluetoothSig.reliability.last_value to read the updated reliability indicator.

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

7.9.1.2.1.3 A2Nreference

SCPI Commands

```
READ:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange:LENergy:LE1M:A2NReference
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange:LENergy:LE1M:A2NReference
CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange:LENergy:LE1M:A2NReference
```

class A2Nreference

A2Nreference commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Trans_Packets: float: decimal No. of transmitted packets Range: 0 to 200E+6
- Rec_Events: float: No parameter help available
- Rec_Events_Val_Crc: float: No parameter help available
- No_Valid_Iqp_Airs: enums.ResultStatus2: decimal No. of valid measurements Range: 0 to 100E+3
- Valid_Iq_Pairs_Rec: enums.ResultStatus2: float Percentage of received valid IQ sample pairs related to all received IQ pairs Range: 0 % to 100 % , Unit: %
- Mean_Amplitude: enums.ResultStatus2: float Mean amplitude Range: 0 to 1 , Unit: dBFS
- Max_Amplitude: enums.ResultStatus2: float Max amplitude Range: 0 to 1 , Unit: dBFS
- Rec_Events_Cnt: enums.ResultStatus2: decimal No. of received events
- Pro_Events_Cnt: enums.ResultStatus2: decimal No. of processed events
- Pro_Events_Val_Crc: enums.ResultStatus2: No parameter help available
- Nos_Pos_Amp: enums.ResultStatus2: No parameter help available

class FetchStruct

Response structure. Fields:

- Trans_Packets: int: decimal No. of transmitted packets Range: 0 to 200E+6
- Rec_Events: float: No parameter help available
- Rec_Events_Val_Crc: int: No parameter help available
- No_Valid_Iqp_Airs: int: decimal No. of valid measurements Range: 0 to 100E+3

- Valid_Iq_Pairs_Rec: float: float Percentage of received valid IQ sample pairs related to all received IQ pairs Range: 0 % to 100 % , Unit: %
- Mean_Amplitude: float: float Mean amplitude Range: 0 to 1 , Unit: dBFS
- Max_Amplitude: float: float Max amplitude Range: 0 to 1 , Unit: dBFS
- Rec_Events_Cnt: int: decimal No. of received events
- Pro_Events_Cnt: int: decimal No. of processed events
- Pro_Events_Val_Crc: int: No parameter help available
- Nos_Pos_Amp: int: No parameter help available

class ReadStruct

Response structure. Fields:

- Trans_Packets: float: decimal No. of transmitted packets Range: 0 to 200E+6
- Rec_Events: float: No parameter help available
- Rec_Events_Val_Crc: float: No parameter help available
- No_Valid_Iqp_Airs: float: decimal No. of valid measurements Range: 0 to 100E+3
- Valid_Iq_Pairs_Rec: float: float Percentage of received valid IQ sample pairs related to all received IQ pairs Range: 0 % to 100 % , Unit: %
- Mean_Amplitude: float: float Mean amplitude Range: 0 to 1 , Unit: dBFS
- Max_Amplitude: float: float Max amplitude Range: 0 to 1 , Unit: dBFS
- Rec_Events_Cnt: int: decimal No. of received events
- Pro_Events_Cnt: int: decimal No. of processed events
- Pro_Events_Val_Crc: int: No parameter help available
- Nof_Pos_Amp: int: decimal No. of possible amplitude measurements determined by CTE length, slot length and number of antennas

calculate() → CalculateStruct

```
# SCPI: CALCulate:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:LENergy:LE1M:A2NReference
value: CalculateStruct = driver.rxQuality.iqDrange.lowEnergy.le1M.a2Nreference.
↪calculate()
```

Returns the results of IQ dynamic range Rx measurement for the specified antenna. Commands for uncoded LE 1M PHY (..:LE1M. .) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second non-reference antenna (...:A1NReference) , and optional third and fourth non-reference antennas are available. The 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 RsCmwBluetoothSig.reliability.last_value to read the updated reliability indicator.

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

fetch() → FetchStruct

```
# SCPI: FETCh:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:LENergy:LE1M:A2NReference
value: FetchStruct = driver.rxQuality.iqDrange.lowEnergy.le1M.a2Nreference.
↪fetch()
```

Returns the results of IQ dynamic range Rx measurement for the specified antenna. Commands for uncoded LE 1M PHY (..:LE1M. .) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second non-reference antenna (...:A1NReference) , and optional third and fourth non-reference antennas are available. The 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 RsCmwBluetoothSig.reliability.last_value to read the updated reliability indicator.

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

read() → ReadStruct

```
# SCPI: READ:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:LENergy:LE1M:A2NReference
value: ReadStruct = driver.rxQuality.iqDrange.lowEnergy.le1M.a2Nreference.read()
```

Returns the results of IQ dynamic range Rx measurement for the specified antenna. Commands for uncoded LE 1M PHY (..:LE1M. .) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second non-reference antenna (...:A1NReference) , and optional third and fourth non-reference antennas are available. The 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 RsCmwBluetoothSig.reliability.last_value to read the updated reliability indicator.

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

7.9.1.2.1.4 A3Nreference

SCPI Commands

```
READ:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange:LENergy:LE1M:A3NReference
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange:LENergy:LE1M:A3NReference
CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange:LENergy:LE1M:A3NReference
```

class A3Nreference

A3Nreference commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Trans_Packets: float: decimal No. of transmitted packets Range: 0 to 200E+6
- Rec_Events: float: No parameter help available
- Rec_Events_Val_Crc: float: No parameter help available
- No_Valid_Iqp_Airs: enums.ResultStatus2: decimal No. of valid measurements Range: 0 to 100E+3
- Valid_Iq_Pairs_Rec: enums.ResultStatus2: float Percentage of received valid IQ sample pairs related to all received IQ pairs Range: 0 % to 100 % , Unit: %
- Mean_Amplitude: enums.ResultStatus2: float Mean amplitude Range: 0 to 1 , Unit: dBFS

- Max_Amplitude: enums.ResultStatus2: float Max amplitude Range: 0 to 1 , Unit: dBFS
- Rec_Events_Cnt: enums.ResultStatus2: decimal No. of received events
- Pro_Events_Cnt: enums.ResultStatus2: decimal No. of processed events
- Pro_Events_Val_Crc: enums.ResultStatus2: No parameter help available
- Nos_Pos_Amp: enums.ResultStatus2: No parameter help available

class FetchStruct

Response structure. Fields:

- Trans_Packets: int: decimal No. of transmitted packets Range: 0 to 200E+6
- Rec_Events: float: No parameter help available
- Rec_Events_Val_Crc: int: No parameter help available
- No_Valid_Iqp_Airs: int: decimal No. of valid measurements Range: 0 to 100E+3
- Valid_Iq_Pairs_Rec: float: float Percentage of received valid IQ sample pairs related to all received IQ pairs Range: 0 % to 100 % , Unit: %
- Mean_Amplitude: float: float Mean amplitude Range: 0 to 1 , Unit: dBFS
- Max_Amplitude: float: float Max amplitude Range: 0 to 1 , Unit: dBFS
- Rec_Events_Cnt: int: decimal No. of received events
- Pro_Events_Cnt: int: decimal No. of processed events
- Pro_Events_Val_Crc: int: No parameter help available
- Nof_Pos_Amp: int: decimal No. of possible amplitude measurements determined by CTE length, slot length and number of antennas

class ReadStruct

Response structure. Fields:

- Trans_Packets: float: decimal No. of transmitted packets Range: 0 to 200E+6
- Rec_Events: float: No parameter help available
- Rec_Events_Val_Crc: float: No parameter help available
- No_Valid_Iqp_Airs: float: decimal No. of valid measurements Range: 0 to 100E+3
- Valid_Iq_Pairs_Rec: float: float Percentage of received valid IQ sample pairs related to all received IQ pairs Range: 0 % to 100 % , Unit: %
- Mean_Amplitude: float: float Mean amplitude Range: 0 to 1 , Unit: dBFS
- Max_Amplitude: float: float Max amplitude Range: 0 to 1 , Unit: dBFS
- Rec_Events_Cnt: int: decimal No. of received events
- Pro_Events_Cnt: int: decimal No. of processed events
- Pro_Events_Val_Crc: int: No parameter help available
- Nof_Pos_Amp: int: decimal No. of possible amplitude measurements determined by CTE length, slot length and number of antennas

calculate() → CalculateStruct

```
# SCPI: CALCulate:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:LENergy:LE1M:A3NReference
value: CalculateStruct = driver.rxQuality.iqDrange.lowEnergy.le1M.a3Nreference.
↪calculate()
```

Returns the results of IQ dynamic range Rx measurement for the specified antenna. Commands for uncoded LE 1M PHY (..:LE1M. .) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second non-reference antenna (...:A1NReference) , and optional third and fourth non-reference antennas are available. The 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 RsCmwBluetoothSig.reliability.last_value to read the updated reliability indicator.

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

fetch() → FetchStruct

```
# SCPI: FETCh:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:LENergy:LE1M:A3NReference
value: FetchStruct = driver.rxQuality.iqDrange.lowEnergy.le1M.a3Nreference.
↪fetch()
```

Returns the results of IQ dynamic range Rx measurement for the specified antenna. Commands for uncoded LE 1M PHY (..:LE1M. .) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second non-reference antenna (...:A1NReference) , and optional third and fourth non-reference antennas are available. The 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 RsCmwBluetoothSig.reliability.last_value to read the updated reliability indicator.

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

read() → ReadStruct

```
# SCPI: READ:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:LENergy:LE1M:A3NReference
value: ReadStruct = driver.rxQuality.iqDrange.lowEnergy.le1M.a3Nreference.read()
```

Returns the results of IQ dynamic range Rx measurement for the specified antenna. Commands for uncoded LE 1M PHY (..:LE1M. .) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second non-reference antenna (...:A1NReference) , and optional third and fourth non-reference antennas are available. The 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 RsCmwBluetoothSig.reliability.last_value to read the updated reliability indicator.

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

7.9.1.2.2 Le2M

class Le2M

Le2M commands group definition. 12 total commands, 4 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.rxQuality.iqDrange.lowEnergy.le2M.clone()
```

Subgroups

7.9.1.2.2.1 A0Reference

SCPI Commands

```
READ:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange:LENergy:LE2M:A0Reference
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange:LENergy:LE2M:A0Reference
CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange:LENergy:LE2M:A0Reference
```

class A0Reference

A0Reference commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Trans_Packets: float: decimal No. of transmitted packets Range: 0 to 200E+6
- Rec_Events: float: No parameter help available
- Rec_Events_Val_Crc: float: No parameter help available
- No_Valid_Iqp_Airs: float: decimal No. of valid measurements Range: 0 to 100E+3
- Valid_Iq_Pairs_Rec: float: float Percentage of received valid IQ sample pairs related to all received IQ pairs Range: 0 % to 100 % , Unit: %
- Mean_Amplitude: enums.ResultStatus2: float Mean amplitude Range: 0 to 1 , Unit: dBFS
- Max_Amplitude: enums.ResultStatus2: float Max amplitude Range: 0 to 1 , Unit: dBFS
- Rec_Events_Cnt: enums.ResultStatus2: decimal No. of received events
- Pro_Events_Cnt: enums.ResultStatus2: decimal No. of processed events
- Pro_Events_Val_Crc: enums.ResultStatus2: No parameter help available
- Nof_Pos_Amp: enums.ResultStatus2: decimal No. of possible amplitude measurements determined by CTE length, slot length and number of antennas

class ResultData

Response structure. Fields:

- Trans_Packets: int: decimal No. of transmitted packets Range: 0 to 200E+6
- Rec_Events: float: No parameter help available
- Rec_Events_Val_Crc: int: No parameter help available

- No_Valid_Iqp_Airs: int: decimal No. of valid measurements Range: 0 to 100E+3
- Valid_Iq_Pairs_Rec: float: float Percentage of received valid IQ sample pairs related to all received IQ pairs Range: 0 % to 100 % , Unit: %
- Mean_Amplitude: float: float Mean amplitude Range: 0 to 1 , Unit: dBFS
- Max_Amplitude: float: float Max amplitude Range: 0 to 1 , Unit: dBFS
- Rec_Events_Cnt: int: decimal No. of received events
- Pro_Events_Cnt: int: decimal No. of processed events
- Pro_Events_Val_Crc: int: No parameter help available
- Nof_Pos_Amp: int: decimal No. of possible amplitude measurements determined by CTE length, slot length and number of antennas

calculate() → CalculateStruct

```
# SCPI: CALCulate:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:LENergy:LE2M:A0Reference
value: CalculateStruct = driver.rxQuality.iqDrange.lowEnergy.le2M.a0Reference.
↪calculate()
```

Returns the results of IQ dynamic range Rx measurement for the specified antenna. Commands for uncoded LE 1M PHY (..:LE1M. .) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second non-reference antenna (...:A1NReference) , and optional third and fourth non-reference antennas are available. The 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 RsCmwBluetoothSig.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:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:LENergy:LE2M:A0Reference
value: ResultData = driver.rxQuality.iqDrange.lowEnergy.le2M.a0Reference.fetch()
```

Returns the results of IQ dynamic range Rx measurement for the specified antenna. Commands for uncoded LE 1M PHY (..:LE1M. .) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second non-reference antenna (...:A1NReference) , and optional third and fourth non-reference antennas are available. The 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 RsCmwBluetoothSig.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:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:LENergy:LE2M:A0Reference
value: ResultData = driver.rxQuality.iqDrange.lowEnergy.le2M.a0Reference.read()
```

Returns the results of IQ dynamic range Rx measurement for the specified antenna. Commands for uncoded LE 1M PHY (..:LE1M. .) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna

(. . . :A0Reference) , mandatory second non-reference antenna (. . . :A1NReference) , and optional third and fourth non-reference antennas are available. The 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 RsCmwBluetoothSig.reliability.last_value to read the updated reliability indicator.

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

7.9.1.2.2.2 A1Nreference

SCPI Commands

```
READ:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange:LENergy:LE2M:A1NReference
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange:LENergy:LE2M:A1NReference
CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange:LENergy:LE2M:A1NReference
```

class A1Nreference

A1Nreference commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Trans_Packets: float: decimal No. of transmitted packets Range: 0 to 200E+6
- Rec_Events: float: No parameter help available
- Rec_Events_Val_Crc: float: No parameter help available
- No_Valid_Iqp_Airs: float: decimal No. of valid measurements Range: 0 to 100E+3
- Valid_Iq_Pairs_Rec: float: float Percentage of received valid IQ sample pairs related to all received IQ pairs Range: 0 % to 100 % , Unit: %
- Mean_Amplitude: enums.ResultStatus2: float Mean amplitude Range: 0 to 1 , Unit: dBFS
- Max_Amplitude: enums.ResultStatus2: float Max amplitude Range: 0 to 1 , Unit: dBFS
- Rec_Events_Cnt: enums.ResultStatus2: decimal No. of received events
- Pro_Events_Cnt: enums.ResultStatus2: decimal No. of processed events
- Pro_Events_Val_Crc: enums.ResultStatus2: No parameter help available
- Nof_Pos_Amp: enums.ResultStatus2: decimal No. of possible amplitude measurements determined by CTE length, slot length and number of antennas

class ResultData

Response structure. Fields:

- Trans_Packets: int: decimal No. of transmitted packets Range: 0 to 200E+6
- Rec_Events: float: No parameter help available
- Rec_Events_Val_Crc: int: No parameter help available
- No_Valid_Iqp_Airs: int: decimal No. of valid measurements Range: 0 to 100E+3
- Valid_Iq_Pairs_Rec: float: float Percentage of received valid IQ sample pairs related to all received IQ pairs Range: 0 % to 100 % , Unit: %
- Mean_Amplitude: float: float Mean amplitude Range: 0 to 1 , Unit: dBFS
- Max_Amplitude: float: float Max amplitude Range: 0 to 1 , Unit: dBFS

- Rec_Events_Cnt: int: decimal No. of received events
- Pro_Events_Cnt: int: decimal No. of processed events
- Pro_Events_Val_Crc: int: No parameter help available
- Nof_Pos_Amp: int: decimal No. of possible amplitude measurements determined by CTE length, slot length and number of antennas

calculate() → CalculateStruct

```
# SCPI: CALCulate:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:LENergy:LE2M:A1NReference
value: CalculateStruct = driver.rxQuality.iqDrange.lowEnergy.le2M.a1Nreference.
↪calculate()
```

Returns the results of IQ dynamic range Rx measurement for the specified antenna. Commands for uncoded LE 1M PHY (..:LE1M. .) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second non-reference antenna (...:A1NReference) , and optional third and fourth non-reference antennas are available. The 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 RsCmwBluetoothSig.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:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:LENergy:LE2M:A1NReference
value: ResultData = driver.rxQuality.iqDrange.lowEnergy.le2M.a1Nreference.
↪fetch()
```

Returns the results of IQ dynamic range Rx measurement for the specified antenna. Commands for uncoded LE 1M PHY (..:LE1M. .) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second non-reference antenna (...:A1NReference) , and optional third and fourth non-reference antennas are available. The 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 RsCmwBluetoothSig.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:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:LENergy:LE2M:A1NReference
value: ResultData = driver.rxQuality.iqDrange.lowEnergy.le2M.a1Nreference.read()
```

Returns the results of IQ dynamic range Rx measurement for the specified antenna. Commands for uncoded LE 1M PHY (..:LE1M. .) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second non-reference antenna (...:A1NReference) , and optional third and fourth non-reference antennas are available. The 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 RsCmwBluetoothSig.reliability.last_value to read the updated reliability indicator.

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

7.9.1.2.2.3 A2Nreference

SCPI Commands

```
READ:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange:LENergy:LE2M:A2NReference
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange:LENergy:LE2M:A2NReference
CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange:LENergy:LE2M:A2NReference
```

class A2Nreference

A2Nreference commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Trans_Packets: float: decimal No. of transmitted packets Range: 0 to 200E+6
- Rec_Events: float: No parameter help available
- Rec_Events_Val_Crc: float: No parameter help available
- No_Valid_Iqp_Airs: float: decimal No. of valid measurements Range: 0 to 100E+3
- Valid_Iq_Pairs_Rec: float: float Percentage of received valid IQ sample pairs related to all received IQ pairs Range: 0 % to 100 % , Unit: %
- Mean_Amplitude: enums.ResultStatus2: float Mean amplitude Range: 0 to 1 , Unit: dBFS
- Max_Amplitude: enums.ResultStatus2: float Max amplitude Range: 0 to 1 , Unit: dBFS
- Rec_Events_Cnt: enums.ResultStatus2: decimal No. of received events
- Pro_Events_Cnt: enums.ResultStatus2: decimal No. of processed events
- Pro_Events_Val_Crc: enums.ResultStatus2: No parameter help available
- Nof_Pos_Amp: enums.ResultStatus2: decimal No. of possible amplitude measurements determined by CTE length, slot length and number of antennas

class FetchStruct

Response structure. Fields:

- Trans_Packets: int: decimal No. of transmitted packets Range: 0 to 200E+6
- Rec_Events: float: No parameter help available
- Rec_Events_Val_Crc: int: No parameter help available
- No_Valid_Iqp_Airs: int: decimal No. of valid measurements Range: 0 to 100E+3
- Valid_Iq_Pairs_Rec: float: float Percentage of received valid IQ sample pairs related to all received IQ pairs Range: 0 % to 100 % , Unit: %
- Mean_Amplitude: float: float Mean amplitude Range: 0 to 1 , Unit: dBFS
- Max_Amplitude: float: float Max amplitude Range: 0 to 1 , Unit: dBFS
- Rec_Events_Cnt: int: decimal No. of received events
- Pro_Events_Cnt: int: decimal No. of processed events
- Pro_Events_Val_Crc: int: No parameter help available
- Nos_Pos_Amp: int: No parameter help available

class ReadStruct

Response structure. Fields:

- Trans_Packets: int: decimal No. of transmitted packets Range: 0 to 200E+6
- Rec_Events: float: No parameter help available
- Rec_Events_Val_Crc: int: No parameter help available
- No_Valid_Iqp_Airs: int: decimal No. of valid measurements Range: 0 to 100E+3
- Valid_Iq_Pairs_Rec: float: float Percentage of received valid IQ sample pairs related to all received IQ pairs Range: 0 % to 100 % , Unit: %
- Mean_Amplitude: float: float Mean amplitude Range: 0 to 1 , Unit: dBFS
- Max_Amplitude: float: float Max amplitude Range: 0 to 1 , Unit: dBFS
- Rec_Events_Cnt: int: decimal No. of received events
- Pro_Events_Cnt: int: decimal No. of processed events
- Pro_Events_Val_Crc: int: No parameter help available
- Nof_Pos_Amp: int: decimal No. of possible amplitude measurements determined by CTE length, slot length and number of antennas

calculate() → CalculateStruct

```
# SCPI: CALCulate:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:LENergy:LE2M:A2NReference
value: CalculateStruct = driver.rxQuality.iqDrange.lowEnergy.le2M.a2Nreference.
↪calculate()
```

Returns the results of IQ dynamic range Rx measurement for the specified antenna. Commands for uncoded LE 1M PHY (..:LE1M. .) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second non-reference antenna (...:A1NReference) , and optional third and fourth non-reference antennas are available. The 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 RsCmwBluetoothSig.reliability.last_value to read the updated reliability indicator.

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

fetch() → FetchStruct

```
# SCPI: FETCh:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:LENergy:LE2M:A2NReference
value: FetchStruct = driver.rxQuality.iqDrange.lowEnergy.le2M.a2Nreference.
↪fetch()
```

Returns the results of IQ dynamic range Rx measurement for the specified antenna. Commands for uncoded LE 1M PHY (..:LE1M. .) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second non-reference antenna (...:A1NReference) , and optional third and fourth non-reference antennas are available. The 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 RsCmwBluetoothSig.reliability.last_value to read the updated reliability indicator.

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

read() → ReadStruct

```
# SCPI: READ:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:LENergy:LE2M:A2NReference
value: ReadStruct = driver.rxQuality.iqDrange.lowEnergy.le2M.a2Nreference.read()
```

Returns the results of IQ dynamic range Rx measurement for the specified antenna. Commands for uncoded LE 1M PHY (..:LE1M. .) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second non-reference antenna (...:A1NReference) , and optional third and fourth non-reference antennas are available. The 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 RsCmwBluetoothSig.reliability.last_value to read the updated reliability indicator.

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

7.9.1.2.2.4 A3Nreference

SCPI Commands

```
READ:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange:LENergy:LE2M:A3NReference
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange:LENergy:LE2M:A3NReference
CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange:LENergy:LE2M:A3NReference
```

class A3Nreference

A3Nreference commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Trans_Packets: float: decimal No. of transmitted packets Range: 0 to 200E+6
- Rec_Events: float: No parameter help available
- Rec_Events_Val_Crc: float: No parameter help available
- No_Valid_Iqp_Airs: float: decimal No. of valid measurements Range: 0 to 100E+3
- Valid_Iq_Pairs_Rec: float: float Percentage of received valid IQ sample pairs related to all received IQ pairs Range: 0 % to 100 % , Unit: %
- Mean_Amplitude: enums.ResultStatus2: float Mean amplitude Range: 0 to 1 , Unit: dBFS
- Max_Amplitude: enums.ResultStatus2: float Max amplitude Range: 0 to 1 , Unit: dBFS
- Rec_Events_Cnt: enums.ResultStatus2: decimal No. of received events
- Pro_Events_Cnt: enums.ResultStatus2: decimal No. of processed events
- Pro_Events_Val_Crc: enums.ResultStatus2: No parameter help available
- Nof_Pos_Amp: enums.ResultStatus2: decimal No. of possible amplitude measurements determined by CTE length, slot length and number of antennas

class ResultData

Response structure. Fields:

- Trans_Packets: int: decimal No. of transmitted packets Range: 0 to 200E+6
- Rec_Events: float: No parameter help available
- Rec_Events_Val_Crc: int: No parameter help available
- No_Valid_Iqp_Airs: int: decimal No. of valid measurements Range: 0 to 100E+3

- Valid_Iq_Pairs_Rec: float: float Percentage of received valid IQ sample pairs related to all received IQ pairs Range: 0 % to 100 % , Unit: %
- Mean_Amplitude: float: float Mean amplitude Range: 0 to 1 , Unit: dBFS
- Max_Amplitude: float: float Max amplitude Range: 0 to 1 , Unit: dBFS
- Rec_Events_Cnt: int: decimal No. of received events
- Pro_Events_Cnt: int: decimal No. of processed events
- Pro_Events_Val_Crc: int: No parameter help available
- Nof_Pos_Amp: int: decimal No. of possible amplitude measurements determined by CTE length, slot length and number of antennas

calculate() → CalculateStruct

```
# SCPI: CALCulate:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:LENergy:LE2M:A3NReference
value: CalculateStruct = driver.rxQuality.iqDrange.lowEnergy.le2M.a3Nreference.
↪calculate()
```

Returns the results of IQ dynamic range Rx measurement for the specified antenna. Commands for uncoded LE 1M PHY (..:LE1M. .) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second non-reference antenna (...:A1NReference) , and optional third and fourth non-reference antennas are available. The 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 RsCmwBluetoothSig.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:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:LENergy:LE2M:A3NReference
value: ResultData = driver.rxQuality.iqDrange.lowEnergy.le2M.a3Nreference.
↪fetch()
```

Returns the results of IQ dynamic range Rx measurement for the specified antenna. Commands for uncoded LE 1M PHY (..:LE1M. .) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna (...:A0Reference) , mandatory second non-reference antenna (...:A1NReference) , and optional third and fourth non-reference antennas are available. The 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 RsCmwBluetoothSig.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:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:LENergy:LE2M:A3NReference
value: ResultData = driver.rxQuality.iqDrange.lowEnergy.le2M.a3Nreference.read()
```

Returns the results of IQ dynamic range Rx measurement for the specified antenna. Commands for uncoded LE 1M PHY (..:LE1M. .) and LE 2M PHY (..:LE2M..) are available. Commands for reference antenna

(. . . :A0Reference) , mandatory second non-reference antenna (. . . :A1NReference) , and optional third and fourth non-reference antennas are available. The 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 RsCmwBluetoothSig.reliability.last_value to read the updated reliability indicator.

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

7.9.1.3 AntMeanAmp

class AntMeanAmp

AntMeanAmp commands group definition. 6 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.9.1.3.1 LowEnergy

class LowEnergy

LowEnergy commands group definition. 6 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.rxQuality.iqDrange.antMeanAmp.lowEnergy.clone()
```

Subgroups

7.9.1.3.1.1 Le1M

SCPI Commands

```
READ:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange:ANTMeanamp:LENergy:LE1M
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange:ANTMeanamp:LENergy:LE1M
CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange:ANTMeanamp:LENergy:LE1M
```

class Le1M

Le1M commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal See ‘Reliability Indicator’
- Ant_3_Minus_2: float: float MeanAmplitudeANT3 - MeanAmplitudeANT2 Range: -256 to 128
- Ant_2_Minus_Ref: float: float MeanAmplitudeANT2 - MeanAmplitudeANT0 Range: -256 to 128

- Ant_Ref_Minus_1: float: float MeanAmplitudeANT0 - MeanAmplitudeANT1 Range: -256 to 128

class ResultData

Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'
- Ant_3_Minus_2: float: float MeanAmplitudeANT3 - MeanAmplitudeANT2 Range: -256 to 128
- Ant_2_Minus_Ref: float: float MeanAmplitudeANT2 - MeanAmplitudeANT0 Range: -256 to 128
- Ant_Ref_Minus_1: float: float MeanAmplitudeANT0 - MeanAmplitudeANT1 Range: -256 to 128

calculate() → CalculateStruct

```
# SCPI: CALCulate:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:ANTMeanamp:LENergy:LE1M
value: CalculateStruct = driver.rxQuality.iqDrange.antMeanAmp.lowEnergy.le1M.
↪calculate()
```

Returns the results of IQ dynamic range Rx measurement for the antenna mean amplitude differences. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available. The 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:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:ANTMeanamp:LENergy:LE1M
value: ResultData = driver.rxQuality.iqDrange.antMeanAmp.lowEnergy.le1M.fetch()
```

Returns the results of IQ dynamic range Rx measurement for the antenna mean amplitude differences. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available. The 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:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:ANTMeanamp:LENergy:LE1M
value: ResultData = driver.rxQuality.iqDrange.antMeanAmp.lowEnergy.le1M.read()
```

Returns the results of IQ dynamic range Rx measurement for the antenna mean amplitude differences. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available. The 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.

7.9.1.3.1.2 Le2M

SCPI Commands

```
READ:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange:ANTMeanamp:LENergy:LE2M
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange:ANTMeanamp:LENergy:LE2M
CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:IQDRange:ANTMeanamp:LENergy:LE2M
```

class Le2M

Le2M commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'
- Ant_3_Minus_2: float: float MeanAmplitudeANT3 - MeanAmplitudeANT2 Range: -256 to 128
- Ant_2_Minus_Ref: float: float MeanAmplitudeANT2 - MeanAmplitudeANT0 Range: -256 to 128
- Ant_Ref_Minus_1: float: float MeanAmplitudeANT0 - MeanAmplitudeANT1 Range: -256 to 128

class ResultData

Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'
- Ant_3_Minus_2: float: float MeanAmplitudeANT3 - MeanAmplitudeANT2 Range: -256 to 128
- Ant_2_Minus_Ref: float: float MeanAmplitudeANT2 - MeanAmplitudeANT0 Range: -256 to 128
- Ant_Ref_Minus_1: float: float MeanAmplitudeANT0 - MeanAmplitudeANT1 Range: -256 to 128

calculate() → CalculateStruct

```
# SCPI: CALCulate:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:ANTMeanamp:LENergy:LE2M
value: CalculateStruct = driver.rxQuality.iqDrange.antMeanAmp.lowEnergy.le2M.
↪calculate()
```

Returns the results of IQ dynamic range Rx measurement for the antenna mean amplitude differences. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available. The 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:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:ANTMeanamp:LENergy:LE2M
value: ResultData = driver.rxQuality.iqDrange.antMeanAmp.lowEnergy.le2M.fetch()
```

Returns the results of IQ dynamic range Rx measurement for the antenna mean amplitude differences. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available. The 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:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQDRange:ANTMeanamp:LENergy:LE2M
value: ResultData = driver.rxQuality.iqDrange.antMeanAmp.lowEnergy.le2M.read()
```

Returns the results of IQ dynamic range Rx measurement for the antenna mean amplitude differences. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available. The 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.

7.9.2 IqCoherency

SCPI Commands

```
INITiate:BLUetooth:SIGNaling<Instance>:RXQuality:IQCoherency
STOP:BLUetooth:SIGNaling<Instance>:RXQuality:IQCoherency
ABORt:BLUetooth:SIGNaling<Instance>:RXQuality:IQCoherency
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:IQCoherency
```

class IqCoherency

IqCoherency commands group definition. 29 total commands, 2 Sub-groups, 4 group commands

abort() → None

```
# SCPI: ABORt:BLUetooth:SIGNaling<Instance>:RXQuality:IQCoherency
driver.rxQuality.iqCoherency.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.

abort_with_opc() → None

```
# SCPI: ABORt:BLUetooth:SIGNaling<Instance>:RXQuality:IQCoherency
driver.rxQuality.iqCoherency.abort_with_opc()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:
```

(continues on next page)

(continued from previous page)

```
    - 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 abort, but waits for the operation to complete before continuing further. Use the RsCmwBluetoothSig.utilities.opc_timeout_set() to set the timeout value.

fetch() → RsCmwBluetoothSig.enums.ResourceState

```
# SCPI: FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:IQCoherency
value: enums.ResourceState = driver.rxQuality.iqCoherency.fetch()
```

No command help available

return meas_status: No help available

initiate() → None

```
# SCPI: INITiate:BLUetooth:SIGNaling<Instance>:RXQuality:IQCoherency
driver.rxQuality.iqCoherency.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.

initiate_with_opc() → None

```
# SCPI: INITiate:BLUetooth:SIGNaling<Instance>:RXQuality:IQCoherency
driver.rxQuality.iqCoherency.initiate_with_opc()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
```

(continues on next page)

(continued from previous page)

```
    - 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 initiate, but waits for the operation to complete before continuing further. Use the RsCmwBluetoothSig.utilities.opc_timeout_set() to set the timeout value.

stop() → None

```
# SCPI: STOP:BLUetooth:SIGNaling<Instance>:RXQuality:IQCoherency
driver.rxQuality.iqCoherency.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() → None

```
# SCPI: STOP:BLUetooth:SIGNaling<Instance>:RXQuality:IQCoherency
driver.rxQuality.iqCoherency.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 RsCmwBluetoothSig.utilities.opc_timeout_set() to set the timeout value.

Cloning the Group

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

Subgroups

7.9.2.1 State

SCPI Commands

```
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:IQCoherency:STATe
```

class State

State commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch() → RsCmwBluetoothSig.enums.ResourceState

```
# SCPI: FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:IQCoherency:STATe
value: enums.ResourceState = driver.rxQuality.iqCoherency.state.fetch()
```

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.

return meas_status: OFF | RDY | RUN OFF: measurement switched off, no resources allocated, no results available (when entered after ABORt...) RDY: measurement has been terminated, valid results can be available RUN: measurement running (after INITiate..., READ...) , synchronization pending or adjusted, resources active or queued

7.9.2.2 LowEnergy

class LowEnergy

LowEnergy commands group definition. 24 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.9.2.2.1 Le1M

class Le1M

Le1M commands group definition. 12 total commands, 4 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.rxQuality.iqCoherency.lowEnergy.le1M.clone()
```

Subgroups

7.9.2.2.1.1 A0Reference

SCPI Commands

```
READ:BLUetooth:SIGNaling<Instance>:RXQuality:IQCoherency:LENergy:LE1M:A0Reference
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:IQCoherency:LENergy:LE1M:A0Reference
CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:IQCoherency:LENergy:LE1M:A0Reference
```

class A0Reference

A0Reference commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'
- Trans_Packets: float: decimal No. of transmitted packets Range: 0 to 200E+6
- Rec_Events: float: No parameter help available
- Rec_Events_Val_Crc: float: No parameter help available
- No_Valid_Iqp_Airs: enums.ResultStatus2: decimal No. of received valid IQ sample pairs
- Valid_Iq_Pairs_Rec: enums.ResultStatus2: float Percentage of received valid IQ sample pairs related to all received IQ pairs
- Mean_Phase_Diff: enums.ResultStatus2: float Mean current phase difference Unit: rad/
- Rec_Events_Cnt: enums.ResultStatus2: decimal No. of received events
- Pro_Events_Cnt: enums.ResultStatus2: decimal No. of processed events
- Pro_Events_Val_Crc: enums.ResultStatus2: No parameter help available
- Nof_Pos_Pha: enums.ResultStatus2: decimal No. of possible phase measurements determined by CTE length, slot length and number of antennas
- Mrpd_Avg: enums.ResultStatus2: float Mean reference phase deviation (RPD) - average value
- Mrpd_Min: enums.ResultStatus2: float Mean RPD - minimum value
- Mrpd_Max: enums.ResultStatus2: float Mean RPD - maximum value

class ResultData

Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'
- Trans_Packets: int: decimal No. of transmitted packets Range: 0 to 200E+6
- Rec_Events: float: No parameter help available
- Rec_Events_Val_Crc: int: No parameter help available

- No_Valid_Iqp_Airs: int: decimal No. of received valid IQ sample pairs
- Valid_Iq_Pairs_Rec: float: float Percentage of received valid IQ sample pairs related to all received IQ pairs
- Mean_Phase_Diff: float: float Mean current phase difference Unit: rad/
- Rec_Events_Cnt: int: decimal No. of received events
- Pro_Events_Cnt: int: decimal No. of processed events
- Pro_Events_Val_Crc: int: No parameter help available
- Nof_Pos_Pha: int: decimal No. of possible phase measurements determined by CTE length, slot length and number of antennas
- Mrpd_Avg: float: float Mean reference phase deviation (RPD) - average value
- Mrpd_Min: float: float Mean RPD - minimum value
- Mrpd_Max: float: float Mean RPD - maximum value

calculate() → CalculateStruct

```
# SCPI: CALCulate:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LENergy:LE1M:A0Reference
value: CalculateStruct = driver.rxQuality.iqCoherency.lowEnergy.le1M.
↪a0Reference.calculate()
```

Return the results of IQ samples coherency RPD measurements for the reference antenna. Commands for uncoded LE 1M PHY (.. :LE1M..) and LE 2M PHY (..:LE2M..) are available. The 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:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LENergy:LE1M:A0Reference
value: ResultData = driver.rxQuality.iqCoherency.lowEnergy.le1M.a0Reference.
↪fetch()
```

Return the results of IQ samples coherency RPD measurements for the reference antenna. Commands for uncoded LE 1M PHY (.. :LE1M..) and LE 2M PHY (..:LE2M..) are available. The 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:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LENergy:LE1M:A0Reference
value: ResultData = driver.rxQuality.iqCoherency.lowEnergy.le1M.a0Reference.
↪read()
```

Return the results of IQ samples coherency RPD measurements for the reference antenna. Commands for uncoded LE 1M PHY (.. :LE1M..) and LE 2M PHY (..:LE2M..) are available. The 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.

7.9.2.2.1.2 A1Nreference

SCPI Commands

```
READ:BLUetooth:SIGNaling<Instance>:RXQuality:IQCoherency:LENergy:LE1M:A1NReference
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:IQCoherency:LENergy:LE1M:A1NReference
CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:IQCoherency:LENergy:LE1M:A1NReference
```

class A1Nreference

A1Nreference commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Trans_Packets: float: decimal No. of transmitted packets Range: 0 to 200E+6
- Rec_Events: float: No parameter help available
- Rec_Events_Val_Crc: float: No parameter help available
- No_Valid_Iqp_Airs: enums.ResultStatus2: decimal No. of received valid IQ sample pairs
- Valid_Iq_Pairs_Rec: enums.ResultStatus2: float Percentage of received valid IQ sample pairs related to all received IQ pairs
- Mean_Phase_Diff: enums.ResultStatus2: No parameter help available
- P_95_Mean_Ph_Diff: enums.ResultStatus2: No parameter help available
- Rec_Events_Cnt: enums.ResultStatus2: decimal No. of received events
- Pro_Events_Cnt: enums.ResultStatus2: decimal No. of processed events
- Pro_Events_Val_Crc: enums.ResultStatus2: No parameter help available
- Nof_Pos_Pha: enums.ResultStatus2: decimal No. of possible phase measurements determined by CTE length, slot length and number of antennas

class ResultData

Response structure. Fields:

- Trans_Packets: int: decimal No. of transmitted packets Range: 0 to 200E+6
- Rec_Events: float: No parameter help available
- Rec_Events_Val_Crc: int: No parameter help available
- No_Valid_Iqp_Airs: int: decimal No. of received valid IQ sample pairs
- Valid_Iq_Pairs_Rec: float: float Percentage of received valid IQ sample pairs related to all received IQ pairs
- Mean_Phase_Diff: float: No parameter help available
- P_95_Mean_Ph_Diff: float: No parameter help available
- Rec_Events_Cnt: int: decimal No. of received events
- Pro_Events_Cnt: int: decimal No. of processed events

- Pro_Events_Val_Crc: int: No parameter help available
- Nof_Pos_Pha: int: decimal No. of possible phase measurements determined by CTE length, slot length and number of antennas

calculate() → CalculateStruct

```
# SCPI: CALCulate:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LENergy:LE1M:A1NReference
value: CalculateStruct = driver.rxQuality.iqCoherency.lowEnergy.le1M.
↪a1Nreference.calculate()
```

Returns the results of IQ samples coherency RP(m) measurement for the specified non-reference antenna. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for the mandatory non-reference antenna (. ..:A1NReference) , and optional third (. . . :A2NReference) and fourth (. . . :A3NReference) antennas are available. The 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 RsCmwBluetoothSig.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:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LENergy:LE1M:A1NReference
value: ResultData = driver.rxQuality.iqCoherency.lowEnergy.le1M.a1Nreference.
↪fetch()
```

Returns the results of IQ samples coherency RP(m) measurement for the specified non-reference antenna. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for the mandatory non-reference antenna (. ..:A1NReference) , and optional third (. . . :A2NReference) and fourth (. . . :A3NReference) antennas are available. The 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 RsCmwBluetoothSig.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:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LENergy:LE1M:A1NReference
value: ResultData = driver.rxQuality.iqCoherency.lowEnergy.le1M.a1Nreference.
↪read()
```

Returns the results of IQ samples coherency RP(m) measurement for the specified non-reference antenna. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for the mandatory non-reference antenna (. ..:A1NReference) , and optional third (. . . :A2NReference) and fourth (. . . :A3NReference) antennas are available. The 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 RsCmwBluetoothSig.reliability.last_value to read the updated reliability indicator.

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

7.9.2.2.1.3 A2Nreference

SCPI Commands

```
READ:BLUetooth:SIGNaling<Instance>:RXQuality:IQCoherency:LENergy:LE1M:A2NReference
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:IQCoherency:LENergy:LE1M:A2NReference
CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:IQCoherency:LENergy:LE1M:A2NReference
```

class A2Nreference

A2Nreference commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Trans_Packets: float: decimal No. of transmitted packets Range: 0 to 200E+6
- Rec_Events: float: No parameter help available
- Rec_Events_Val_Crc: float: No parameter help available
- No_Valid_Iqp_Airs: enums.ResultStatus2: decimal No. of received valid IQ sample pairs
- Valid_Iq_Pairs_Rec: enums.ResultStatus2: float Percentage of received valid IQ sample pairs related to all received IQ pairs
- Mean_Phase_Diff: enums.ResultStatus2: No parameter help available
- P_95_Mean_Ph_Diff: enums.ResultStatus2: No parameter help available
- Rec_Events_Cnt: enums.ResultStatus2: decimal No. of received events
- Pro_Events_Cnt: enums.ResultStatus2: decimal No. of processed events
- Pro_Events_Val_Crc: enums.ResultStatus2: No parameter help available
- Nof_Pos_Pha: enums.ResultStatus2: decimal No. of possible phase measurements determined by CTE length, slot length and number of antennas

class ResultData

Response structure. Fields:

- Trans_Packets: int: decimal No. of transmitted packets Range: 0 to 200E+6
- Rec_Events: float: No parameter help available
- Rec_Events_Val_Crc: int: No parameter help available
- No_Valid_Iqp_Airs: int: decimal No. of received valid IQ sample pairs
- Valid_Iq_Pairs_Rec: float: float Percentage of received valid IQ sample pairs related to all received IQ pairs
- Mean_Phase_Diff: float: No parameter help available
- P_95_Mean_Ph_Diff: float: No parameter help available
- Rec_Events_Cnt: int: decimal No. of received events
- Pro_Events_Cnt: int: decimal No. of processed events
- Pro_Events_Val_Crc: int: No parameter help available

- Nof_Pos_Pha: int: decimal No. of possible phase measurements determined by CTE length, slot length and number of antennas

calculate() → CalculateStruct

```
# SCPI: CALCulate:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LENergy:LE1M:A2NReference
value: CalculateStruct = driver.rxQuality.iqCoherency.lowEnergy.le1M.
↪a2Nreference.calculate()
```

Returns the results of IQ samples coherency RP(m) measurement for the specified non-reference antenna. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for the mandatory non-reference antenna (. ..:A1NReference) , and optional third (. . . :A2NReference) and fourth (. . . :A3NReference) antennas are available. The 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 RsCmwBluetoothSig.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:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LENergy:LE1M:A2NReference
value: ResultData = driver.rxQuality.iqCoherency.lowEnergy.le1M.a2Nreference.
↪fetch()
```

Returns the results of IQ samples coherency RP(m) measurement for the specified non-reference antenna. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for the mandatory non-reference antenna (. ..:A1NReference) , and optional third (. . . :A2NReference) and fourth (. . . :A3NReference) antennas are available. The 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 RsCmwBluetoothSig.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:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LENergy:LE1M:A2NReference
value: ResultData = driver.rxQuality.iqCoherency.lowEnergy.le1M.a2Nreference.
↪read()
```

Returns the results of IQ samples coherency RP(m) measurement for the specified non-reference antenna. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for the mandatory non-reference antenna (. ..:A1NReference) , and optional third (. . . :A2NReference) and fourth (. . . :A3NReference) antennas are available. The 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 RsCmwBluetoothSig.reliability.last_value to read the updated reliability indicator.

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

7.9.2.2.1.4 A3Nreference

SCPI Commands

```
READ:BLUetooth:SIGNaling<Instance>:RXQuality:IQCoherency:LENergy:LE1M:A3NReference
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:IQCoherency:LENergy:LE1M:A3NReference
CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:IQCoherency:LENergy:LE1M:A3NReference
```

class A3Nreference

A3Nreference commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Trans_Packets: float: decimal No. of transmitted packets Range: 0 to 200E+6
- Rec_Events: float: No parameter help available
- Rec_Events_Val_Crc: float: No parameter help available
- No_Valid_Iqp_Airs: enums.ResultStatus2: decimal No. of received valid IQ sample pairs
- Valid_Iq_Pairs_Rec: enums.ResultStatus2: float Percentage of received valid IQ sample pairs related to all received IQ pairs
- Mean_Phase_Diff: enums.ResultStatus2: No parameter help available
- P_95_Mean_Ph_Diff: enums.ResultStatus2: No parameter help available
- Rec_Events_Cnt: enums.ResultStatus2: decimal No. of received events
- Pro_Events_Cnt: enums.ResultStatus2: decimal No. of processed events
- Pro_Events_Val_Crc: enums.ResultStatus2: No parameter help available
- Nof_Pos_Pha: enums.ResultStatus2: decimal No. of possible phase measurements determined by CTE length, slot length and number of antennas

class ResultData

Response structure. Fields:

- Trans_Packets: int: decimal No. of transmitted packets Range: 0 to 200E+6
- Rec_Events: float: No parameter help available
- Rec_Events_Val_Crc: int: No parameter help available
- No_Valid_Iqp_Airs: int: decimal No. of received valid IQ sample pairs
- Valid_Iq_Pairs_Rec: float: float Percentage of received valid IQ sample pairs related to all received IQ pairs
- Mean_Phase_Diff: float: No parameter help available
- P_95_Mean_Ph_Diff: float: No parameter help available
- Rec_Events_Cnt: int: decimal No. of received events
- Pro_Events_Cnt: int: decimal No. of processed events
- Pro_Events_Val_Crc: int: No parameter help available
- Nof_Pos_Pha: int: decimal No. of possible phase measurements determined by CTE length, slot length and number of antennas

calculate() → CalculateStruct

```
# SCPI: CALCulate:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LENergy:LE1M:A3NReference
value: CalculateStruct = driver.rxQuality.iqCoherency.lowEnergy.le1M.
↪a3Nreference.calculate()
```

Returns the results of IQ samples coherency RP(m) measurement for the specified non-reference antenna. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for the mandatory non-reference antenna (. ..:A1NReference) , and optional third (. . . :A2NReference) and fourth (. . . :A3NReference) antennas are available. The 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 RsCmwBluetoothSig.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:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LENergy:LE1M:A3NReference
value: ResultData = driver.rxQuality.iqCoherency.lowEnergy.le1M.a3Nreference.
↪fetch()
```

Returns the results of IQ samples coherency RP(m) measurement for the specified non-reference antenna. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for the mandatory non-reference antenna (. ..:A1NReference) , and optional third (. . . :A2NReference) and fourth (. . . :A3NReference) antennas are available. The 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 RsCmwBluetoothSig.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:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LENergy:LE1M:A3NReference
value: ResultData = driver.rxQuality.iqCoherency.lowEnergy.le1M.a3Nreference.
↪read()
```

Returns the results of IQ samples coherency RP(m) measurement for the specified non-reference antenna. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for the mandatory non-reference antenna (. ..:A1NReference) , and optional third (. . . :A2NReference) and fourth (. . . :A3NReference) antennas are available. The 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 RsCmwBluetoothSig.reliability.last_value to read the updated reliability indicator.

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

7.9.2.2.2 Le2M

class Le2M
Le2M commands group definition. 12 total commands, 4 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.rxQuality.iqCoherency.lowEnergy.le2M.clone()
```

Subgroups

7.9.2.2.2.1 A0Reference

SCPI Commands

```
READ:BLUetooth:SIGNaling<Instance>:RXQuality:IQCoherency:LENergy:LE2M:A0Reference
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:IQCoherency:LENergy:LE2M:A0Reference
CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:IQCoherency:LENergy:LE2M:A0Reference
```

class A0Reference
A0Reference commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class CalculateStruct
Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'
- Trans_Packets: float: decimal No. of transmitted packets Range: 0 to 200E+6
- Rec_Events: float: No parameter help available
- Rec_Events_Val_Crc: float: No parameter help available
- No_Valid_Iqp_Airs: enums.ResultStatus2: decimal No. of received valid IQ sample pairs
- Valid_Iq_Pairs_Rec: enums.ResultStatus2: float Percentage of received valid IQ sample pairs related to all received IQ pairs
- Mean_Phase_Diff: enums.ResultStatus2: float Mean current phase difference Unit: rad/
- Rec_Events_Cnt: enums.ResultStatus2: decimal No. of received events
- Pro_Events_Cnt: enums.ResultStatus2: decimal No. of processed events
- Pro_Events_Val_Crc: enums.ResultStatus2: No parameter help available
- Nof_Pos_Pha: enums.ResultStatus2: decimal No. of possible phase measurements determined by CTE length, slot length and number of antennas
- Mrpd_Avg: enums.ResultStatus2: float Mean reference phase deviation (RPD) - average value
- Mrpd_Min: enums.ResultStatus2: float Mean RPD - minimum value
- Mrpd_Max: enums.ResultStatus2: float Mean RPD - maximum value

class ResultData
Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'
- Trans_Packets: int: decimal No. of transmitted packets Range: 0 to 200E+6
- Rec_Events: float: No parameter help available
- Rec_Events_Val_Crc: int: No parameter help available
- No_Valid_Iqp_Airs: int: decimal No. of received valid IQ sample pairs
- Valid_Iq_Pairs_Rec: float: float Percentage of received valid IQ sample pairs related to all received IQ pairs
- Mean_Phase_Diff: float: float Mean current phase difference Unit: rad/
- Rec_Events_Cnt: int: decimal No. of received events
- Pro_Events_Cnt: int: decimal No. of processed events
- Pro_Events_Val_Crc: int: No parameter help available
- Nof_Pos_Pha: int: decimal No. of possible phase measurements determined by CTE length, slot length and number of antennas
- Mrpd_Avg: float: float Mean reference phase deviation (RPD) - average value
- Mrpd_Min: float: float Mean RPD - minimum value
- Mrpd_Max: float: float Mean RPD - maximum value

calculate() → CalculateStruct

```
# SCPI: CALCulate:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LENergy:LE2M:A0Reference
value: CalculateStruct = driver.rxQuality.iqCoherency.lowEnergy.le2M.
↪a0Reference.calculate()
```

Return the results of IQ samples coherency RPD measurements for the reference antenna. Commands for uncoded LE 1M PHY (.. :LE1M..) and LE 2M PHY (..:LE2M..) are available. The 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:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LENergy:LE2M:A0Reference
value: ResultData = driver.rxQuality.iqCoherency.lowEnergy.le2M.a0Reference.
↪fetch()
```

Return the results of IQ samples coherency RPD measurements for the reference antenna. Commands for uncoded LE 1M PHY (.. :LE1M..) and LE 2M PHY (..:LE2M..) are available. The 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:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LENergy:LE2M:A0Reference
value: ResultData = driver.rxQuality.iqCoherency.lowEnergy.le2M.a0Reference.
↪read()
```

Return the results of IQ samples coherency RPD measurements for the reference antenna. Commands for uncoded LE 1M PHY (.. :LE1M..) and LE 2M PHY (..:LE2M..) are available. The 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.

7.9.2.2.2.2 A1Nreference

SCPI Commands

```
READ:BLUetooth:SIGNaling<Instance>:RXQuality:IQCoherency:LENergy:LE2M:A1NReference
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:IQCoherency:LENergy:LE2M:A1NReference
CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:IQCoherency:LENergy:LE2M:A1NReference
```

class A1Nreference

A1Nreference commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Trans_Packets: float: decimal No. of transmitted packets Range: 0 to 200E+6
- Rec_Events: float: No parameter help available
- Rec_Events_Val_Crc: float: No parameter help available
- No_Valid_Iqp_Airs: enums.ResultStatus2: decimal No. of received valid IQ sample pairs
- Valid_Iq_Pairs_Rec: enums.ResultStatus2: float Percentage of received valid IQ sample pairs related to all received IQ pairs
- Mean_Phase_Diff: enums.ResultStatus2: No parameter help available
- P_95_Mean_Ph_Diff: enums.ResultStatus2: No parameter help available
- Rec_Events_Cnt: enums.ResultStatus2: decimal No. of received events
- Pro_Events_Cnt: enums.ResultStatus2: decimal No. of processed events
- Pro_Events_Val_Crc: enums.ResultStatus2: No parameter help available
- Nof_Pos_Pha: enums.ResultStatus2: decimal No. of possible phase measurements determined by CTE length, slot length and number of antennas

class ResultData

Response structure. Fields:

- Trans_Packets: int: decimal No. of transmitted packets Range: 0 to 200E+6
- Rec_Events: float: No parameter help available
- Rec_Events_Val_Crc: int: No parameter help available
- No_Valid_Iqp_Airs: int: decimal No. of received valid IQ sample pairs

- Valid_Iq_Pairs_Rec: float: float Percentage of received valid IQ sample pairs related to all received IQ pairs
- Mean_Phase_Diff: float: No parameter help available
- P_95_Mean_Ph_Diff: float: No parameter help available
- Rec_Events_Cnt: int: decimal No. of received events
- Pro_Events_Cnt: int: decimal No. of processed events
- Pro_Events_Val_Crc: int: No parameter help available
- Nof_Pos_Pha: int: decimal No. of possible phase measurements determined by CTE length, slot length and number of antennas

calculate() → CalculateStruct

```
# SCPI: CALCulate:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LENergy:LE2M:A1NReference
value: CalculateStruct = driver.rxQuality.iqCoherency.lowEnergy.le2M.
↪a1Nreference.calculate()
```

Returns the results of IQ samples coherency RP(m) measurement for the specified non-reference antenna. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for the mandatory non-reference antenna (. ..:A1NReference) , and optional third (. . . :A2NReference) and fourth (. . . :A3NReference) antennas are available. The 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 RsCmwBluetoothSig.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:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LENergy:LE2M:A1NReference
value: ResultData = driver.rxQuality.iqCoherency.lowEnergy.le2M.a1Nreference.
↪fetch()
```

Returns the results of IQ samples coherency RP(m) measurement for the specified non-reference antenna. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for the mandatory non-reference antenna (. ..:A1NReference) , and optional third (. . . :A2NReference) and fourth (. . . :A3NReference) antennas are available. The 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 RsCmwBluetoothSig.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:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LENergy:LE2M:A1NReference
value: ResultData = driver.rxQuality.iqCoherency.lowEnergy.le2M.a1Nreference.
↪read()
```

Returns the results of IQ samples coherency RP(m) measurement for the specified non-reference antenna. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for the mandatory non-reference antenna (. ..:A1NReference) , and optional third (. . . :A2NReference) and fourth (. . . :A3NReference) antennas are available. The 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 RsCmwBluetoothSig.reliability.last_value to read the updated reliability indicator.

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

7.9.2.2.2.3 A2Nreference

SCPI Commands

```
READ:BLUetooth:SIGNaling<Instance>:RXQuality:IQCoherency:LENergy:LE2M:A2NReference
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:IQCoherency:LENergy:LE2M:A2NReference
CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:IQCoherency:LENergy:LE2M:A2NReference
```

class A2Nreference

A2Nreference commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Trans_Packets: float: decimal No. of transmitted packets Range: 0 to 200E+6
- Rec_Events: float: No parameter help available
- Rec_Events_Val_Crc: float: No parameter help available
- No_Valid_Iqp_Airs: enums.ResultStatus2: decimal No. of received valid IQ sample pairs
- Valid_Iq_Pairs_Rec: enums.ResultStatus2: float Percentage of received valid IQ sample pairs related to all received IQ pairs
- Mean_Phase_Diff: enums.ResultStatus2: No parameter help available
- P_95_Mean_Ph_Diff: enums.ResultStatus2: No parameter help available
- Rec_Events_Cnt: float: decimal No. of received events
- Pro_Events_Cnt: enums.ResultStatus2: decimal No. of processed events
- Pro_Events_Val_Crc: enums.ResultStatus2: No parameter help available
- Nof_Pos_Pha: enums.ResultStatus2: decimal No. of possible phase measurements determined by CTE length, slot length and number of antennas

class ResultData

Response structure. Fields:

- Trans_Packets: int: decimal No. of transmitted packets Range: 0 to 200E+6
- Rec_Events: float: No parameter help available
- Rec_Events_Val_Crc: int: No parameter help available
- No_Valid_Iqp_Airs: int: decimal No. of received valid IQ sample pairs
- Valid_Iq_Pairs_Rec: float: float Percentage of received valid IQ sample pairs related to all received IQ pairs

- Mean_Phase_Diff: float: No parameter help available
- P_95_Mean_Ph_Diff: float: No parameter help available
- Rec_Events_Cnt: int: decimal No. of received events
- Pro_Events_Cnt: int: decimal No. of processed events
- Pro_Events_Val_Crc: int: No parameter help available
- Nof_Pos_Pha: int: decimal No. of possible phase measurements determined by CTE length, slot length and number of antennas

calculate() → CalculateStruct

```
# SCPI: CALCulate:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LENergy:LE2M:A2NReference
value: CalculateStruct = driver.rxQuality.iqCoherency.lowEnergy.le2M.
↪a2Nreference.calculate()
```

Returns the results of IQ samples coherency RP(m) measurement for the specified non-reference antenna. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for the mandatory non-reference antenna (. ..:A1NReference) , and optional third (. . . :A2NReference) and fourth (. . . :A3NReference) antennas are available. The 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 RsCmwBluetoothSig.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:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LENergy:LE2M:A2NReference
value: ResultData = driver.rxQuality.iqCoherency.lowEnergy.le2M.a2Nreference.
↪fetch()
```

Returns the results of IQ samples coherency RP(m) measurement for the specified non-reference antenna. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for the mandatory non-reference antenna (. ..:A1NReference) , and optional third (. . . :A2NReference) and fourth (. . . :A3NReference) antennas are available. The 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 RsCmwBluetoothSig.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:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LENergy:LE2M:A2NReference
value: ResultData = driver.rxQuality.iqCoherency.lowEnergy.le2M.a2Nreference.
↪read()
```

Returns the results of IQ samples coherency RP(m) measurement for the specified non-reference antenna. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands

for the mandatory non-reference antenna (. ..:A1NReference) , and optional third (. . . :A2NReference) and fourth (. . . :A3NReference) antennas are available. The 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 RsCmwBluetoothSig.reliability.last_value to read the updated reliability indicator.

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

7.9.2.2.2.4 A3Nreference

SCPI Commands

```
READ:BLUetooth:SIGNaling<Instance>:RXQuality:IQCoherency:LENergy:LE2M:A3NReference
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:IQCoherency:LENergy:LE2M:A3NReference
CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:IQCoherency:LENergy:LE2M:A3NReference
```

class A3Nreference

A3Nreference commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Trans_Packets: float: decimal No. of transmitted packets Range: 0 to 200E+6
- Rec_Events: float: No parameter help available
- Rec_Events_Val_Crc: float: No parameter help available
- No_Valid_Iqp_Airs: enums.ResultStatus2: decimal No. of received valid IQ sample pairs
- Valid_Iq_Pairs_Rec: enums.ResultStatus2: float Percentage of received valid IQ sample pairs related to all received IQ pairs
- Mean_Phase_Diff: enums.ResultStatus2: No parameter help available
- P_95_Mean_Ph_Diff: enums.ResultStatus2: No parameter help available
- Rec_Events_Cnt: enums.ResultStatus2: decimal No. of received events
- Pro_Events_Cnt: enums.ResultStatus2: decimal No. of processed events
- Pro_Events_Val_Crc: enums.ResultStatus2: No parameter help available
- Nof_Pos_Pha: enums.ResultStatus2: decimal No. of possible phase measurements determined by CTE length, slot length and number of antennas

class ResultData

Response structure. Fields:

- Trans_Packets: int: decimal No. of transmitted packets Range: 0 to 200E+6
- Rec_Events: float: No parameter help available
- Rec_Events_Val_Crc: int: No parameter help available
- No_Valid_Iqp_Airs: int: decimal No. of received valid IQ sample pairs
- Valid_Iq_Pairs_Rec: float: float Percentage of received valid IQ sample pairs related to all received IQ pairs
- Mean_Phase_Diff: float: No parameter help available
- P_95_Mean_Ph_Diff: float: No parameter help available

- Rec_Events_Cnt: int: decimal No. of received events
- Pro_Events_Cnt: int: decimal No. of processed events
- Pro_Events_Val_Crc: int: No parameter help available
- Nof_Pos_Pha: int: decimal No. of possible phase measurements determined by CTE length, slot length and number of antennas

calculate() → CalculateStruct

```
# SCPI: CALCulate:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LENergy:LE2M:A3NReference
value: CalculateStruct = driver.rxQuality.iqCoherency.lowEnergy.le2M.
↪a3Nreference.calculate()
```

Returns the results of IQ samples coherency RP(m) measurement for the specified non-reference antenna. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for the mandatory non-reference antenna (. ..:A1NReference) , and optional third (. . . :A2NReference) and fourth (. . . :A3NReference) antennas are available. The 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 RsCmwBluetoothSig.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:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LENergy:LE2M:A3NReference
value: ResultData = driver.rxQuality.iqCoherency.lowEnergy.le2M.a3Nreference.
↪fetch()
```

Returns the results of IQ samples coherency RP(m) measurement for the specified non-reference antenna. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for the mandatory non-reference antenna (. ..:A1NReference) , and optional third (. . . :A2NReference) and fourth (. . . :A3NReference) antennas are available. The 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 RsCmwBluetoothSig.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:BLUetooth:SIGNaling<Instance>
↪:RXQuality:IQCoherency:LENergy:LE2M:A3NReference
value: ResultData = driver.rxQuality.iqCoherency.lowEnergy.le2M.a3Nreference.
↪read()
```

Returns the results of IQ samples coherency RP(m) measurement for the specified non-reference antenna. Commands for uncoded LE 1M PHY (..:LE1M..) and LE 2M PHY (..:LE2M..) are available. Commands for the mandatory non-reference antenna (. ..:A1NReference) , and optional third (. . . :A2NReference) and fourth (. . . :A3NReference) antennas are available. The 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 RsCmwBluetoothSig.reliability.last_value to read the updated reliability indicator.

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

7.9.3 Search

class Search

Search commands group definition. 39 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.9.3.1 Per

SCPI Commands

```
INITiate:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER
STOP:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER
ABORt:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER
```

class Per

Per commands group definition. 31 total commands, 4 Sub-groups, 3 group commands

abort() → None

```
# SCPI: ABORt:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER
driver.rxQuality.search.per.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.

abort_with_opc() → None

```
# SCPI: ABORt:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER
driver.rxQuality.search.per.abort_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 abort, but waits for the operation to complete before continuing further. Use the RsCmwBluetoothSig.utilities.opc_timeout_set() to set the timeout value.

initiate() → None

```
# SCPI: INITiate:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER
driver.rxQuality.search.per.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.

initiate_with_opc() → None

```
# SCPI: INITiate:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER
driver.rxQuality.search.per.initiate_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.
```

(continues on next page)

(continued from previous page)

Use FETCh...STATe? to query the current measurement state.

Same as initiate, but waits for the operation to complete before continuing further. Use the RsCmwBluetoothSig.utilities.opc_timeout_set() to set the timeout value.

stop() → None

```
# SCPI: STOP:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER
driver.rxQuality.search.per.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() → None

```
# SCPI: STOP:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER
driver.rxQuality.search.per.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 RsCmwBluetoothSig.utilities.opc_timeout_set() to set the timeout value.

Cloning the Group

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

Subgroups

7.9.3.1.1 State

SCPI Commands

```
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER:STATe
```

class State

State commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch() → RsCmwBluetoothSig.enums.ResourceState

```
# SCPI: FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER:STATe
value: enums.ResourceState = driver.rxQuality.search.per.state.fetch()
```

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.

return meas_status: OFF | RDY | RUN OFF: measurement switched off, no resources allocated, no results available (when entered after ABORt...) RDY: measurement has been terminated, valid results can be available RUN: measurement running (after INITiate..., READ...) , synchronization pending or adjusted, resources active or queued

7.9.3.1.2 LowEnergy

class LowEnergy

LowEnergy commands group definition. 9 total commands, 3 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.rxQuality.search.per.lowEnergy.clone()
```

Subgroups

7.9.3.1.2.1 Le1M

SCPI Commands

```
READ:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER:LENergy:LE1M
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER:LENergy:LE1M
CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER:LENergy:LE1M
```

class Le1M

Le1M commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'
- Per: float: float Packet error rate Range: 0 % to 100 %, Unit: %
- Packets_Received: float: decimal Number of correct packets received and reported by the EUT Range: 0 to 30E+3
- Search_Result: float: float TX level of the R&S CMW resulting in the configured PER search limit Range: -999 dBm to 0 dBm, Unit: dBm

class ResultData

Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'
- Per: float: float Packet error rate Range: 0 % to 100 %, Unit: %
- Packets_Received: int: decimal Number of correct packets received and reported by the EUT Range: 0 to 30E+3
- Search_Result: float: float TX level of the R&S CMW resulting in the configured PER search limit Range: -999 dBm to 0 dBm, Unit: dBm

calculate() → CalculateStruct

```
# SCPI: CALCulate:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:PER:LENergy[:LE1M]
value: CalculateStruct = driver.rxQuality.search.per.lowEnergy.le1M.calculate()
```

Return the results of PER search RX measurement for LE direct test mode and LE connection tests.

INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (..:NMODe:LENergy:LRANge..) are available.

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

fetch() → ResultData

```
# SCPI: FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER:LENergy[:LE1M]
value: ResultData = driver.rxQuality.search.per.lowEnergy.le1M.fetch()
```

Return the results of PER search RX measurement for LE direct test mode and LE connection tests.

INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (..:NMODe:LENergy:LRANge..) are available.

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

read() → ResultData

```
# SCPI: READ:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER:LENergy[:LE1M]
value: ResultData = driver.rxQuality.search.per.lowEnergy.le1M.read()
```

Return the results of PER search RX measurement for LE direct test mode and LE connection tests.
INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (..:NMODe:LENergy:LRANge..) are available.

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

7.9.3.1.2.2 Lrange

SCPI Commands

```
READ:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER:LENergy:LRANge
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER:LENergy:LRANge
CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER:LENergy:LRANge
```

class Lrange
Lrange commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class CalculateStruct
Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'
- Per: float: float Packet error rate Range: 0 % to 100 %, Unit: %
- Packets_Received: float: decimal Number of correct packets received and reported by the EUT Range: 0 to 30E+3
- Search_Result: float: float TX level of the R&S CMW resulting in the configured PER search limit Range: -999 dBm to 0 dBm, Unit: dBm

class ResultData
Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'
- Per: float: float Packet error rate Range: 0 % to 100 %, Unit: %

- Packets_Received: int: decimal Number of correct packets received and reported by the EUT Range: 0 to 30E+3
- Search_Result: float: float TX level of the R&S CMW resulting in the configured PER search limit Range: -999 dBm to 0 dBm, Unit: dBm

calculate() → CalculateStruct

```
# SCPI: CALCulate:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:PER:LENergy:LRANge
value: CalculateStruct = driver.rxQuality.search.per.lowEnergy.lrange.
↪calculate()
```

Return the results of PER search RX measurement for LE direct test mode and LE connection tests.
INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (..:NMODe:LENergy:LRANge..) are available.

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

fetch() → ResultData

```
# SCPI: FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER:LENergy:LRANge
value: ResultData = driver.rxQuality.search.per.lowEnergy.lrange.fetch()
```

Return the results of PER search RX measurement for LE direct test mode and LE connection tests.
INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (..:NMODe:LENergy:LRANge..) are available.

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

read() → ResultData

```
# SCPI: READ:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER:LENergy:LRANge
value: ResultData = driver.rxQuality.search.per.lowEnergy.lrange.read()
```

Return the results of PER search RX measurement for LE direct test mode and LE connection tests.
INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.

- LE connection tests (normal mode) : Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (..:NMODe:LENergy:LRANge..) are available.

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

7.9.3.1.2.3 Le2M

SCPI Commands

```
READ:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER:LENergy:LE2M
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER:LENergy:LE2M
CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER:LENergy:LE2M
```

class Le2M

Le2M commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'
- Per: float: float Packet error rate Range: 0 % to 100 %, Unit: %
- Packets_Received: float: decimal Number of correct packets received and reported by the EUT Range: 0 to 30E+3
- Search_Result: float: float TX level of the R&S CMW resulting in the configured PER search limit Range: -999 dBm to 0 dBm, Unit: dBm

class ResultData

Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'
- Per: float: float Packet error rate Range: 0 % to 100 %, Unit: %
- Packets_Received: int: decimal Number of correct packets received and reported by the EUT Range: 0 to 30E+3
- Search_Result: float: float TX level of the R&S CMW resulting in the configured PER search limit Range: -999 dBm to 0 dBm, Unit: dBm

calculate() → CalculateStruct

```
# SCPI: CALCulate:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:PER:LENergy:LE2M
value: CalculateStruct = driver.rxQuality.search.per.lowEnergy.le2M.calculate()
```

Return the results of PER search RX measurement for LE direct test mode and LE connection tests.

INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (..:NMODe:LENergy:LRANge..) are available.

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

fetch() → ResultData

```
# SCPI: FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER:LENergy:LE2M
value: ResultData = driver.rxQuality.search.per.lowEnergy.le2M.fetch()
```

Return the results of PER search RX measurement for LE direct test mode and LE connection tests.
INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (..:NMODe:LENergy:LRANge..) are available.

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

read() → ResultData

```
# SCPI: READ:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER:LENergy:LE2M
value: ResultData = driver.rxQuality.search.per.lowEnergy.le2M.read()
```

Return the results of PER search RX measurement for LE direct test mode and LE connection tests.
INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (..:NMODe:LENergy:LRANge..) are available.

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

7.9.3.1.3 Nmode

class Nmode
Nmode commands group definition. 9 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.rxQuality.search.per.nmode.clone()
```

Subgroups

7.9.3.1.3.1 LowEnergy

class LowEnergy
LowEnergy commands group definition. 9 total commands, 3 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.rxQuality.search.per.nmode.lowEnergy.clone()
```

Subgroups

7.9.3.1.3.2 Le1M

SCPI Commands

```
READ:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER:NMODe:LENergy:LE1M
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER:NMODe:LENergy:LE1M
CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER:NMODe:LENergy:LE1M
```

class Le1M
Le1M commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class CalculateStruct
Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'
- Per: float: float Packet error rate Range: 0 % to 100 %, Unit: %
- Packets_Received: float: decimal Number of correct packets received and reported by the EUT Range: 0 to 30E+3
- Search_Result: float: float TX level of the R&S CMW resulting in the configured PER search limit Range: -999 dBm to 0 dBm, Unit: dBm

class ResultData
Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'
- Per: float: float Packet error rate Range: 0 % to 100 %, Unit: %
- Packets_Received: int: decimal Number of correct packets received and reported by the EUT Range: 0 to 30E+3
- Search_Result: float: float TX level of the R&S CMW resulting in the configured PER search limit Range: -999 dBm to 0 dBm, Unit: dBm

calculate() → CalculateStruct

```
# SCPI: CALCulate:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:PER:NMODe:LENergy:LE1M
value: CalculateStruct = driver.rxQuality.search.per.nmode.lowEnergy.le1M.
↪calculate()
```

Return the results of PER search RX measurement for LE direct test mode and LE connection tests.
INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (..:NMODe:LENergy:LRANge..) are available.

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

fetch() → ResultData

```
# SCPI: FETCh:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:PER:NMODe:LENergy:LE1M
value: ResultData = driver.rxQuality.search.per.nmode.lowEnergy.le1M.fetch()
```

Return the results of PER search RX measurement for LE direct test mode and LE connection tests.
INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (..:NMODe:LENergy:LRANge..) are available.

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

read() → ResultData

```
# SCPI: READ:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:PER:NMODe:LENergy:LE1M
value: ResultData = driver.rxQuality.search.per.nmode.lowEnergy.le1M.read()
```

Return the results of PER search RX measurement for LE direct test mode and LE connection tests.
INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (..:NMODe:LENergy:LRANge..) are available.

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

7.9.3.1.3.3 Le2M

SCPI Commands

```
READ:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER:NMODe:LENergy:LE2M
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER:NMODe:LENergy:LE2M
CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER:NMODe:LENergy:LE2M
```

class Le2M

Le2M commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'
- Per: float: float Packet error rate Range: 0 % to 100 %, Unit: %
- Packets_Received: float: decimal Number of correct packets received and reported by the EUT Range: 0 to 30E+3
- Search_Result: float: float TX level of the R&S CMW resulting in the configured PER search limit Range: -999 dBm to 0 dBm, Unit: dBm

class ResultData

Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'
- Per: float: float Packet error rate Range: 0 % to 100 %, Unit: %
- Packets_Received: int: decimal Number of correct packets received and reported by the EUT Range: 0 to 30E+3
- Search_Result: float: float TX level of the R&S CMW resulting in the configured PER search limit Range: -999 dBm to 0 dBm, Unit: dBm

calculate() → CalculateStruct

```
# SCPI: CALCulate:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:PER:NMODe:LENergy:LE2M
value: CalculateStruct = driver.rxQuality.search.per.nmode.lowEnergy.le2M.
↪calculate()
```

Return the results of PER search RX measurement for LE direct test mode and LE connection tests.

INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (..:NMODe:LENergy:LRANge..) are available.

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

fetch() → ResultData

```
# SCPI: FETCh:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:PER:NMODe:LENergy:LE2M
value: ResultData = driver.rxQuality.search.per.nmode.lowEnergy.le2M.fetch()
```

Return the results of PER search RX measurement for LE direct test mode and LE connection tests.
INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (..:NMODe:LENergy:LRANge..) are available.

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

read() → ResultData

```
# SCPI: READ:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:PER:NMODe:LENergy:LE2M
value: ResultData = driver.rxQuality.search.per.nmode.lowEnergy.le2M.read()
```

Return the results of PER search RX measurement for LE direct test mode and LE connection tests.
INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (..:NMODe:LENergy:LRANge..) are available.

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

7.9.3.1.3.4 Lrange

SCPI Commands

```
READ:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER:NMODe:LENergy:LRANge
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER:NMODe:LENergy:LRANge
CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER:NMODe:LENergy:LRANge
```

class Lrange
Lrange commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class CalculateStruct
Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'
- Per: float: float Packet error rate Range: 0 % to 100 %, Unit: %
- Packets_Received: float: decimal Number of correct packets received and reported by the EUT Range: 0 to 30E+3

- Search_Result: float: float TX level of the R&S CMW resulting in the configured PER search limit Range: -999 dBm to 0 dBm, Unit: dBm

class ResultData

Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'
- Per: float: float Packet error rate Range: 0 % to 100 %, Unit: %
- Packets_Received: int: decimal Number of correct packets received and reported by the EUT Range: 0 to 30E+3
- Search_Result: float: float TX level of the R&S CMW resulting in the configured PER search limit Range: -999 dBm to 0 dBm, Unit: dBm

calculate() → CalculateStruct

```
# SCPI: CALCulate:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:PER:NMODe:LENergy:LRANge
value: CalculateStruct = driver.rxQuality.search.per.nmode.lowEnergy.lrange.
↪calculate()
```

Return the results of PER search RX measurement for LE direct test mode and LE connection tests.

INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (..:NMODe:LENergy:LRANge..) are available.

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

fetch() → ResultData

```
# SCPI: FETCh:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:PER:NMODe:LENergy:LRANge
value: ResultData = driver.rxQuality.search.per.nmode.lowEnergy.lrange.fetch()
```

Return the results of PER search RX measurement for LE direct test mode and LE connection tests.

INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (..:NMODe:LENergy:LRANge..) are available.

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

read() → ResultData

```
# SCPI: READ:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:PER:NMODe:LENergy:LRANge
value: ResultData = driver.rxQuality.search.per.nmode.lowEnergy.lrange.read()
```

Return the results of PER search RX measurement for LE direct test mode and LE connection tests.
INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (..:NMODe:LENergy:LRANge..) are available.

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

7.9.3.1.4 Tmode

class Tmode
Tmode commands group definition. 9 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.rxQuality.search.per.tmode.clone()
```

Subgroups

7.9.3.1.4.1 LowEnergy

class LowEnergy
LowEnergy commands group definition. 9 total commands, 3 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.rxQuality.search.per.tmode.lowEnergy.clone()
```

Subgroups

7.9.3.1.4.2 Le1M

SCPI Commands

```
CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER:TMODe:LENergy:LE1M
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER:TMODe:LENergy:LE1M
READ:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER:TMODe:LENergy:LE1M
```

class Le1M

Le1M commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'
- Per: float: float Packet error rate Range: 0 % to 100 %
- Cor_Packets_Recv: float: decimal Number of correct received packets reported by the EUT Range: 0 to 30E+3
- Search_Result: float: float TX level of the R&S CMW resulting in the configured PER search limit Range: -999 dBm to 0 dBm, Unit: dBm
- Ber: float: float Bit error rate Range: 0 % to 100 %
- Packets_Received: float: decimal Number of received packets detected by the R&S CMW Range: 0 to 30E+3
- Num_Invalid_Crc: enums.ResultStatus2: decimal Number of packets with detected CRC error
- Num_Pattern_Err: enums.ResultStatus2: decimal Number of packets with detected pattern error
- Num_Payload_Err: enums.ResultStatus2: decimal Number of packets with detected payload length error
- Bit_Errors: int: decimal Number of detected bit errors

class ResultData

Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'
- Per: float: float Packet error rate Range: 0 % to 100 %
- Cor_Packets_Recv: int: decimal Number of correct received packets reported by the EUT Range: 0 to 30E+3
- Search_Result: float: float TX level of the R&S CMW resulting in the configured PER search limit Range: -999 dBm to 0 dBm, Unit: dBm
- Ber: float: float Bit error rate Range: 0 % to 100 %
- Packets_Received: int: decimal Number of received packets detected by the R&S CMW Range: 0 to 30E+3
- Num_Invalid_Crc: int: decimal Number of packets with detected CRC error
- Num_Pattern_Err: int: decimal Number of packets with detected pattern error
- Num_Payload_Err: int: decimal Number of packets with detected payload length error
- Bit_Errors: int: decimal Number of detected bit errors

calculate() → CalculateStruct

```
# SCPI: CALCulate:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:PER:TMODe:LENergy:LE1M
value: CalculateStruct = driver.rxQuality.search.per.tmode.lowEnergy.le1M.
↪calculate()
```

Return the results of PER search RX measurement in LE test mode. Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (..:TMODe:LENergy:LRANge..) are available.

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

fetch() → ResultData

```
# SCPI: FETCh:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:PER:TMODe:LENergy:LE1M
value: ResultData = driver.rxQuality.search.per.tmode.lowEnergy.le1M.fetch()
```

Return the results of PER search RX measurement in LE test mode. Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (..:TMODe:LENergy:LRANge..) are available.

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

read() → ResultData

```
# SCPI: READ:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:PER:TMODe:LENergy:LE1M
value: ResultData = driver.rxQuality.search.per.tmode.lowEnergy.le1M.read()
```

Return the results of PER search RX measurement in LE test mode. Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (..:TMODe:LENergy:LRANge..) are available.

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

7.9.3.1.4.3 Le2M

SCPI Commands

```
CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER:TMODe:LENergy:LE2M
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER:TMODe:LENergy:LE2M
READ:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER:TMODe:LENergy:LE2M
```

class Le2M

Le2M commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'
- Per: float: float Packet error rate Range: 0 % to 100 %
- Cor_Packets_Recv: float: decimal Number of correct received packets reported by the EUT Range: 0 to 30E+3

- Search_Result: float: float TX level of the R&S CMW resulting in the configured PER search limit Range: -999 dBm to 0 dBm, Unit: dBm
- Ber: float: float Bit error rate Range: 0 % to 100 %
- Packets_Received: float: decimal Number of received packets detected by the R&S CMW Range: 0 to 30E+3
- Num_Invalid_Crc: enums.ResultStatus2: decimal Number of packets with detected CRC error
- Num_Pattern_Err: enums.ResultStatus2: decimal Number of packets with detected pattern error
- Num_Payload_Err: enums.ResultStatus2: decimal Number of packets with detected payload length error
- Bit_Errors: enums.ResultStatus2: decimal Number of detected bit errors

class ResultData

Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'
- Per: float: float Packet error rate Range: 0 % to 100 %
- Cor_Packets_Recv: int: decimal Number of correct received packets reported by the EUT Range: 0 to 30E+3
- Search_Result: float: float TX level of the R&S CMW resulting in the configured PER search limit Range: -999 dBm to 0 dBm, Unit: dBm
- Ber: float: float Bit error rate Range: 0 % to 100 %
- Packets_Received: int: decimal Number of received packets detected by the R&S CMW Range: 0 to 30E+3
- Num_Invalid_Crc: int: decimal Number of packets with detected CRC error
- Num_Pattern_Err: int: decimal Number of packets with detected pattern error
- Num_Payload_Err: int: decimal Number of packets with detected payload length error
- Bit_Errors: int: decimal Number of detected bit errors

calculate() → CalculateStruct

```
# SCPI: CALCulate:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:PER:TMODe:LENergy:LE2M
value: CalculateStruct = driver.rxQuality.search.per.tmode.lowEnergy.le2M.
↪calculate()
```

Return the results of PER search RX measurement in LE test mode. Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (..:TMODe:LENergy:LRANge..) are available.

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

fetch() → ResultData

```
# SCPI: FETCh:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:PER:TMODe:LENergy:LE2M
value: ResultData = driver.rxQuality.search.per.tmode.lowEnergy.le2M.fetch()
```

Return the results of PER search RX measurement in LE test mode. Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (..:TMODe:LENergy:LRANge..) are available.

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

read() → ResultData

```
# SCPI: READ:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:PER:TMODe:LENergy:LE2M
value: ResultData = driver.rxQuality.search.per.tmode.lowEnergy.le2M.read()
```

Return the results of PER search RX measurement in LE test mode. Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (..:TMODe:LENergy:LRANge..) are available.

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

7.9.3.1.4.4 Lrange

SCPI Commands

```
CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER:TMODe:LENergy:LRANge
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER:TMODe:LENergy:LRANge
READ:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:PER:TMODe:LENergy:LRANge
```

class Lrange

Lrange commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'
- Per: float: float Packet error rate Range: 0 % to 100 %
- Cor_Packets_Recv: float: decimal Number of correct received packets reported by the EUT Range: 0 to 30E+3
- Search_Result: float: float TX level of the R&S CMW resulting in the configured PER search limit Range: -999 dBm to 0 dBm, Unit: dBm
- Ber: float: float Bit error rate Range: 0 % to 100 %
- Packets_Received: float: decimal Number of received packets detected by the R&S CMW Range: 0 to 30E+3
- Num_Invalid_Crc: enums.ResultStatus2: decimal Number of packets with detected CRC error
- Num_Pattern_Err: enums.ResultStatus2: decimal Number of packets with detected pattern error
- Num_Payload_Err: enums.ResultStatus2: decimal Number of packets with detected payload length error
- Bit_Errors: enums.ResultStatus2: decimal Number of detected bit errors

class ResultData

Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'

- Per: float: float Packet error rate Range: 0 % to 100 %
- Cor_Packets_Recv: int: decimal Number of correct received packets reported by the EUT Range: 0 to 30E+3
- Search_Result: float: float TX level of the R&S CMW resulting in the configured PER search limit Range: -999 dBm to 0 dBm, Unit: dBm
- Ber: float: float Bit error rate Range: 0 % to 100 %
- Packets_Received: int: decimal Number of received packets detected by the R&S CMW Range: 0 to 30E+3
- Num_Invalid_Crc: int: decimal Number of packets with detected CRC error
- Num_Pattern_Err: int: decimal Number of packets with detected pattern error
- Num_Payload_Err: int: decimal Number of packets with detected payload length error
- Bit_Errors: int: decimal Number of detected bit errors

calculate() → CalculateStruct

```
# SCPI: CALCulate:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:PER:TMODe:LENergy:LRANge
value: CalculateStruct = driver.rxQuality.search.per.tmode.lowEnergy.lrange.
↪calculate()
```

Return the results of PER search RX measurement in LE test mode. Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (..:TMODe:LENergy:LRANge..) are available.

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

fetch() → ResultData

```
# SCPI: FETCh:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:PER:TMODe:LENergy:LRANge
value: ResultData = driver.rxQuality.search.per.tmode.lowEnergy.lrange.fetch()
```

Return the results of PER search RX measurement in LE test mode. Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (..:TMODe:LENergy:LRANge..) are available.

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

read() → ResultData

```
# SCPI: READ:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:PER:TMODe:LENergy:LRANge
value: ResultData = driver.rxQuality.search.per.tmode.lowEnergy.lrange.read()
```

Return the results of PER search RX measurement in LE test mode. Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M..) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (..:TMODe:LENergy:LRANge..) are available.

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

7.9.3.2 Ber

class Ber
Ber commands group definition. 8 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.9.3.2.1 Bedr

SCPI Commands

```
INITiate:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:BER:BEDR
ABORt:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:BER:BEDR
STOP:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:BER:BEDR
READ:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:BER:BEDR
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:BER:BEDR
CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:BER:BEDR
```

class Bedr
Bedr commands group definition. 6 total commands, 0 Sub-groups, 6 group commands

class CalculateStruct
Response structure. Fields:

- Ber: float: float Bit error rate Range: 0 % to 100 %, Unit: %
- Per: float: float Packet error rate Unit: %
- Bit_Errors: float: decimal Sum of received erroneous data bits Range: 0 to 18.4467440737096E+18
- Missing_Packets: float: float Difference between the number of packets sent and the number of packets received in percentage Unit: %
- Nak: float: float Percentage of packets not acknowledged by the EUT positively Unit: %
- Hec_Errors: float: float Percentage of packets with the bit errors in the header Unit: %
- Crc_Errors: float: float Percentage of packets with the bit errors in the payload Unit: %
- Wrong_Packet_Rate: float: No parameter help available
- Wrong_Payload_Rat: float: No parameter help available
- Packets_Received: float: No parameter help available
- Search_Result: float: float TX level of the R&S CMW resulting in the configured BER search limit Range: -100 dBm to 0 dBm

class ResultData
Response structure. Fields:

- Ber: float: float Bit error rate Range: 0 % to 100 %, Unit: %
- Per: float: float Packet error rate Unit: %

- Bit_Errors: int: decimal Sum of received erroneous data bits Range: 0 to 18.4467440737096E+18
- Missing_Packets: float: float Difference between the number of packets sent and the number of packets received in percentage Unit: %
- Nak: float: float Percentage of packets not acknowledged by the EUT positively Unit: %
- Hec_Errors: float: float Percentage of packets with the bit errors in the header Unit: %
- Crc_Errors: float: float Percentage of packets with the bit errors in the payload Unit: %
- Wrong_Packet_Rate: float: No parameter help available
- Wrong_Payload_Rat: float: No parameter help available
- Packets_Received: int: No parameter help available
- Search_Result: float: float TX level of the R&S CMW resulting in the configured BER search limit Range: -100 dBm to 0 dBm

abort() → None

```
# SCPI: ABORt:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:BER[:BEDR]
driver.rxQuality.search.ber.bedr.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.

abort_with_opc() → None

```
# SCPI: ABORt:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:BER[:BEDR]
driver.rxQuality.search.ber.bedr.abort_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 abort, but waits for the operation to complete before continuing further. Use the RsCmwBluetoothSig.utilities.opc_timeout_set() to set the timeout value.

calculate() → CalculateStruct

```
# SCPI: CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:BER[:BEDR]
value: CalculateStruct = driver.rxQuality.search.ber.bedr.calculate()
```

Return all results of the BR/EDR BER search 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.

Use RsCmwBluetoothSig.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:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:BER[:BEDR]
value: ResultData = driver.rxQuality.search.ber.bedr.fetch()
```

Return all results of the BR/EDR BER search 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.

Use RsCmwBluetoothSig.reliability.last_value to read the updated reliability indicator.

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

initiate() → None

```
# SCPI: INITiate:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:BER[:BEDR]
driver.rxQuality.search.ber.bedr.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.

initiate_with_opc() → None

```
# SCPI: INITiate:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:BER[:BEDR]
driver.rxQuality.search.ber.bedr.initiate_with_opc()
```

(continues on next page)

(continued from previous page)

```
    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 initiate, but waits for the operation to complete before continuing further. Use the RsCmwBluetoothSig.utilities.opc_timeout_set() to set the timeout value.

read() → ResultData

```
# SCPI: READ:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:BER[:BEDR]
value: ResultData = driver.rxQuality.search.ber.bedr.read()
```

Return all results of the BR/EDR BER search 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.

Use RsCmwBluetoothSig.reliability.last_value to read the updated reliability indicator.

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

stop() → None

```
# SCPI: STOP:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:BER[:BEDR]
driver.rxQuality.search.ber.bedr.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() → None

```
# SCPI: STOP:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:BER[:BEDR]
driver.rxQuality.search.ber.bedr.stop_with_opc()
```

(continues on next page)

(continued from previous page)

```
    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 RsCmwBluetoothSig.utilities.opc_timeout_set() to set the timeout value.

7.9.3.2.2 State

class State

State commands group definition. 2 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.rxQuality.search.ber.state.clone()
```

Subgroups

7.9.3.2.2.1 All

class All

All commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.rxQuality.search.ber.state.all.clone()
```

Subgroups

7.9.3.2.2.2 Bedr

SCPI Commands

```
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:BER:STATe:ALL:BEDR
```

class Bedr

Bedr commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Main_State: enums.ResourceState: OFF | RDY | RUN OFF: measurement switched off, no resources allocated, no results available (when entered after STOP...) RDY: measurement has been terminated, valid results can be available RUN: measurement running (after INITiate..., READ...) , synchronization pending or adjusted, resources active or queued
- Sync_State: enums.ResourceState: PEND | ADJ | INV PEND: waiting for resource allocation, adjustment, hardware switching ('pending') ADJ: all necessary adjustments finished, measurement running ('adjusted') INV: not applicable because main_state: OFF or RDY ('invalid')
- Resource_State: enums.ResourceState: QUE | ACT | INV QUE: measurement without resources, no results available ('queued') ACT: resources allocated, acquisition of results in progress but not complete ('active') INV: not applicable because main_state: OFF or RDY ('invalid')

fetch() → FetchStruct

```
# SCPI: FETCh:BLUetooth:SIGNaling<Instance>
↪:RXQuality:SEARch:BER:STATe:ALL[:BEDR]
value: FetchStruct = driver.rxQuality.search.ber.state.all.bedr.fetch()
```

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.

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

7.9.3.2.2.3 Bedr

SCPI Commands

```
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:BER:STATe:BEDR
```

class Bedr

Bedr commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch() → RsCmwBluetoothSig.enums.ResourceState

```
# SCPI: FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:SEARch:BER:STATe[:BEDR]
value: enums.ResourceState = driver.rxQuality.search.ber.state.bedr.fetch()
```

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.

return meas_status: OFF | RDY | RUN OFF: measurement switched off, no resources allocated, no results available (when entered after ABORt...) RDY: measurement has been terminated, valid results can be available RUN: measurement running (after INITiate..., READ...) , synchronization pending or adjusted, resources active or queued

7.9.4 Per

SCPI Commands

```
ABORt:BLUetooth:SIGNaling<Instance>:RXQuality:PER
INITiate:BLUetooth:SIGNaling<Instance>:RXQuality:PER
STOP:BLUetooth:SIGNaling<Instance>:RXQuality:PER
```

class Per

Per commands group definition. 31 total commands, 4 Sub-groups, 3 group commands

abort() → None

```
# SCPI: ABORt:BLUetooth:SIGNaling<Instance>:RXQuality:PER
driver.rxQuality.per.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.

abort_with_opc() → None

```
# SCPI: ABORt:BLUetooth:SIGNaling<Instance>:RXQuality:PER
driver.rxQuality.per.abort_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.
```

(continues on next page)

(continued from previous page)

Use FETCh...STATe? to query the current measurement state.

Same as abort, but waits for the operation to complete before continuing further. Use the RsCmwBluetoothSig.utilities.opc_timeout_set() to set the timeout value.

initiate() → None

```
# SCPI: INITiate:BLUetooth:SIGNaling<Instance>:RXQuality:PER
driver.rxQuality.per.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.

initiate_with_opc() → None

```
# SCPI: INITiate:BLUetooth:SIGNaling<Instance>:RXQuality:PER
driver.rxQuality.per.initiate_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 initiate, but waits for the operation to complete before continuing further. Use the RsCmwBluetoothSig.utilities.opc_timeout_set() to set the timeout value.

stop() → None

```
# SCPI: STOP:BLUetooth:SIGNaling<Instance>:RXQuality:PER
driver.rxQuality.per.stop()
```

(continues on next page)

(continued from previous page)

```
    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() → None

```
# SCPI: STOP:BLUetooth:SIGNaling<Instance>:RXQuality:PER
driver.rxQuality.per.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 RsCmwBluetoothSig.utilities.opc_timeout_set() to set the timeout value.

Cloning the Group

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

Subgroups

7.9.4.1 State

SCPI Commands

```
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:PER:STATe
```

class State

State commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch() → RsCmwBluetoothSig.enums.ResourceState

```
# SCPI: FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:PER:STATe
value: enums.ResourceState = driver.rxQuality.per.state.fetch()
```

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.

return meas_status: OFF | RDY | RUN OFF: measurement switched off, no resources allocated, no results available (when entered after ABORt...) RDY: measurement has been terminated, valid results can be available RUN: measurement running (after INITiate..., READ...) , synchronization pending or adjusted, resources active or queued

7.9.4.2 LowEnergy

class LowEnergy

LowEnergy commands group definition. 9 total commands, 3 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.9.4.2.1 Le1M

SCPI Commands

```
READ:BLUetooth:SIGNaling<Instance>:RXQuality:PER:LENergy:LE1M
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:PER:LENergy:LE1M
CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:PER:LENergy:LE1M
```

class Le1M

Le1M commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'

- Per: float: float Packet error rate Range: 0 % to 100 %, Unit: %
- Packets_Received: float: decimal Number of correct packets received and reported by the EUT. Range: 0 to 30E+3

class ResultData

Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'
- Per: float: float Packet error rate Range: 0 % to 100 %, Unit: %
- Packets_Received: int: decimal Number of correct packets received and reported by the EUT. Range: 0 to 30E+3

calculate() → CalculateStruct

```
# SCPI: CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:PER:LENergy[:LE1M]
value: CalculateStruct = driver.rxQuality.per.lowEnergy.le1M.calculate()
```

Return all results of the signaling LE Rx measurement for LE direct test and LE connection test.

INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (..:NMODe:LENergy:LRANge..) are available.

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

fetch() → ResultData

```
# SCPI: FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:PER:LENergy[:LE1M]
value: ResultData = driver.rxQuality.per.lowEnergy.le1M.fetch()
```

Return all results of the signaling LE Rx measurement for LE direct test and LE connection test.

INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (..:NMODe:LENergy:LRANge..) are available.

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

read() → ResultData

```
# SCPI: READ:BLUetooth:SIGNaling<Instance>:RXQuality:PER:LENergy[:LE1M]
value: ResultData = driver.rxQuality.per.lowEnergy.le1M.read()
```

Return all results of the signaling LE Rx measurement for LE direct test and LE connection test.

INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (..:NMODe:LENergy:LRANge..) are available.

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

7.9.4.2.2 Lrange

SCPI Commands

```
READ:BLUetooth:SIGNaling<Instance>:RXQuality:PER:LENergy:LRANge
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:PER:LENergy:LRANge
CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:PER:LENergy:LRANge
```

class Lrange

Lrange commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'
- Per: float: float Packet error rate Range: 0 % to 100 %, Unit: %
- Packets_Received: float: decimal Number of correct packets received and reported by the EUT. Range: 0 to 30E+3

class ResultData

Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'
- Per: float: float Packet error rate Range: 0 % to 100 %, Unit: %
- Packets_Received: int: decimal Number of correct packets received and reported by the EUT. Range: 0 to 30E+3

calculate() → CalculateStruct

```
# SCPI: CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:PER:LENergy:LRANge
value: CalculateStruct = driver.rxQuality.per.lowEnergy.lrange.calculate()
```

Return all results of the signaling LE Rx measurement for LE direct test and LE connection test.

INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (..:NMODe:LENergy:LRANge..) are available.

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

fetch() → ResultData

```
# SCPI: FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:PER:LENergy:LRANge
value: ResultData = driver.rxQuality.per.lowEnergy.lrange.fetch()
```

Return all results of the signaling LE Rx measurement for LE direct test and LE connection test.
INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (..:NMODe:LENergy:LRANge..) are available.

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

read() → ResultData

```
# SCPI: READ:BLUetooth:SIGNaling<Instance>:RXQuality:PER:LENergy:LRANge
value: ResultData = driver.rxQuality.per.lowEnergy.lrange.read()
```

Return all results of the signaling LE Rx measurement for LE direct test and LE connection test.
INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (..:NMODe:LENergy:LRANge..) are available.

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

7.9.4.2.3 Le2M

SCPI Commands

```
READ:BLUetooth:SIGNaling<Instance>:RXQuality:PER:LENergy:LE2M
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:PER:LENergy:LE2M
CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:PER:LENergy:LE2M
```

class Le2M
Le2M commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class CalculateStruct
Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'
- Per: float: float Packet error rate Range: 0 % to 100 %, Unit: %
- Packets_Received: float: decimal Number of correct packets received and reported by the EUT. Range: 0 to 30E+3

class ResultData
Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'
- Per: float: float Packet error rate Range: 0 % to 100 %, Unit: %
- Packets_Received: int: decimal Number of correct packets received and reported by the EUT. Range: 0 to 30E+3

calculate() → CalculateStruct

```
# SCPI: CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:PER:LENergy:LE2M
value: CalculateStruct = driver.rxQuality.per.lowEnergy.le2M.calculate()
```

Return all results of the signaling LE Rx measurement for LE direct test and LE connection test.
INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (..:NMODe:LENergy:LRANge..) are available.

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

fetch() → ResultData

```
# SCPI: FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:PER:LENergy:LE2M
value: ResultData = driver.rxQuality.per.lowEnergy.le2M.fetch()
```

Return all results of the signaling LE Rx measurement for LE direct test and LE connection test.
INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (..:NMODe:LENergy:LRANge..) are available.

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

read() → ResultData

```
# SCPI: READ:BLUetooth:SIGNaling<Instance>:RXQuality:PER:LENergy:LE2M
value: ResultData = driver.rxQuality.per.lowEnergy.le2M.read()
```

Return all results of the signaling LE Rx measurement for LE direct test and LE connection test.
INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.

- LE connection tests (normal mode) : Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (..:NMODe:LENergy:LRANge..) are available.

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

7.9.4.3 Nmode

class Nmode
Nmode commands group definition. 9 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.9.4.3.1 LowEnergy

class LowEnergy
LowEnergy commands group definition. 9 total commands, 3 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.rxQuality.per.nmode.lowEnergy.clone()
```

Subgroups

7.9.4.3.1.1 Le1M

SCPI Commands

```
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:PER:NMODe:LENergy:LE1M
READ:BLUetooth:SIGNaling<Instance>:RXQuality:PER:NMODe:LENergy:LE1M
CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:PER:NMODe:LENergy:LE1M
```

class Le1M
Le1M commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class CalculateStruct
Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'
- Per: float: float Packet error rate Range: 0 % to 100 %, Unit: %
- Packets_Received: float: decimal Number of correct packets received and reported by the EUT. Range: 0 to 30E+3

class ResultData
Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'
- Per: float: float Packet error rate Range: 0 % to 100 %, Unit: %
- Packets_Received: int: decimal Number of correct packets received and reported by the EUT. Range: 0 to 30E+3

calculate() → CalculateStruct

```
# SCPI: CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:PER:NMODe:LENergy:LE1M
value: CalculateStruct = driver.rxQuality.per.nmode.lowEnergy.le1M.calculate()
```

Return all results of the signaling LE Rx measurement for LE direct test and LE connection test.
INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (..:NMODe:LENergy:LRANge..) are available.

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

fetch() → ResultData

```
# SCPI: FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:PER:NMODe:LENergy:LE1M
value: ResultData = driver.rxQuality.per.nmode.lowEnergy.le1M.fetch()
```

Return all results of the signaling LE Rx measurement for LE direct test and LE connection test.
INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (..:NMODe:LENergy:LRANge..) are available.

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

read() → ResultData

```
# SCPI: READ:BLUetooth:SIGNaling<Instance>:RXQuality:PER:NMODe:LENergy:LE1M
value: ResultData = driver.rxQuality.per.nmode.lowEnergy.le1M.read()
```

Return all results of the signaling LE Rx measurement for LE direct test and LE connection test.
INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.

- LE connection tests (normal mode) : Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (..:NMODe:LENergy:LRANge..) are available.

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

7.9.4.3.1.2 Le2M

SCPI Commands

```
READ:BLUetooth:SIGNaling<Instance>:RXQuality:PER:NMODe:LENergy:LE2M
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:PER:NMODe:LENergy:LE2M
CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:PER:NMODe:LENergy:LE2M
```

class Le2M

Le2M commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'
- Per: float: float Packet error rate Range: 0 % to 100 %, Unit: %
- Packets_Received: float: decimal Number of correct packets received and reported by the EUT. Range: 0 to 30E+3

class ResultData

Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'
- Per: float: float Packet error rate Range: 0 % to 100 %, Unit: %
- Packets_Received: int: decimal Number of correct packets received and reported by the EUT. Range: 0 to 30E+3

calculate() → CalculateStruct

```
# SCPI: CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:PER:NMODe:LENergy:LE2M
value: CalculateStruct = driver.rxQuality.per.nmode.lowEnergy.le2M.calculate()
```

Return all results of the signaling LE Rx measurement for LE direct test and LE connection test.

INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (..:NMODe:LENergy:LRANge..) are available.

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

fetch() → ResultData

```
# SCPI: FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:PER:NMODe:LENergy:LE2M
value: ResultData = driver.rxQuality.per.nmode.lowEnergy.le2M.fetch()
```

Return all results of the signaling LE Rx measurement for LE direct test and LE connection test.
INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (..:NMODe:LENergy:LRANge..) are available.

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

read() → ResultData

```
# SCPI: READ:BLUetooth:SIGNaling<Instance>:RXQuality:PER:NMODe:LENergy:LE2M
value: ResultData = driver.rxQuality.per.nmode.lowEnergy.le2M.read()
```

Return all results of the signaling LE Rx measurement for LE direct test and LE connection test.
INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (..:NMODe:LENergy:LRANge..) are available.

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

7.9.4.3.1.3 Lrange

SCPI Commands

```
READ:BLUetooth:SIGNaling<Instance>:RXQuality:PER:NMODe:LENergy:LRANge
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:PER:NMODe:LENergy:LRANge
CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:PER:NMODe:LENergy:LRANge
```

class Lrange
Lrange commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class CalculateStruct
Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'
- Per: float: float Packet error rate Range: 0 % to 100 %, Unit: %
- Packets_Received: float: decimal Number of correct packets received and reported by the EUT. Range: 0 to 30E+3

class ResultData
Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'
- Per: float: float Packet error rate Range: 0 % to 100 %, Unit: %
- Packets_Received: int: decimal Number of correct packets received and reported by the EUT. Range: 0 to 30E+3

calculate() → CalculateStruct

```
# SCPI: CALCulate:BLUetooth:SIGNaling<Instance>
↪:RXQuality:PER:NMODe:LENergy:LRANge
value: CalculateStruct = driver.rxQuality.per.nmode.lowEnergy.lrange.calculate()
```

Return all results of the signaling LE Rx measurement for LE direct test and LE connection test.
INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (..:NMODe:LENergy:LRANge..) are available.

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

fetch() → ResultData

```
# SCPI: FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:PER:NMODe:LENergy:LRANge
value: ResultData = driver.rxQuality.per.nmode.lowEnergy.lrange.fetch()
```

Return all results of the signaling LE Rx measurement for LE direct test and LE connection test.
INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (..:NMODe:LENergy:LRANge..) are available.

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

read() → ResultData

```
# SCPI: READ:BLUetooth:SIGNaling<Instance>:RXQuality:PER:NMODe:LENergy:LRANge
value: ResultData = driver.rxQuality.per.nmode.lowEnergy.lrange.read()
```

Return all results of the signaling LE Rx measurement for LE direct test and LE connection test.
INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

- LE direct test mode: Commands for uncoded LE 1M PHY (..:LE1M..) , LE 2M PHY (..:LE2M..) , and LE coded PHY (..:LRANge..) are available.
- LE connection tests (normal mode) : Commands for uncoded LE 1M PHY (..:NMODe:LENergy:LE1M..) , LE 2M PHY (..:NMODe:LENergy:LE2M..) , and LE coded PHY (..:NMODe:LENergy:LRANge..) are available.

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

7.9.4.4 Tmode

class Tmode
Tmode commands group definition. 9 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.9.4.4.1 LowEnergy

class LowEnergy
LowEnergy commands group definition. 9 total commands, 3 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.rxQuality.per.tmode.lowEnergy.clone()
```

Subgroups

7.9.4.4.1.1 Le1M

SCPI Commands

```
CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:PER:TMODe:LENergy:LE1M
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:PER:TMODe:LENergy:LE1M
READ:BLUetooth:SIGNaling<Instance>:RXQuality:PER:TMODe:LENergy:LE1M
```

class Le1M
Le1M commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class CalculateStruct
Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'
- Per: float: float Packet error rate Range: 0 % to 100 %, Unit: %
- Cor_Packets_Recv: float: decimal Number of correct received packets reported by the EUT Range: 0 to 30E+3
- Ber: float: float Bit error rate Range: 0 % to 100 %
- Packets_Received: float: decimal Number of received packets detected by the R&S CMW Range: 0 to 30E+3

- Num_Invalid_Crc: enums.ResultStatus2: decimal Number of packets with detected CRC error
- Num_Pattern_Err: enums.ResultStatus2: decimal Number of packets with detected pattern error
- Num_Payload_Err: enums.ResultStatus2: decimal Number of packets with detected payload length error
- Bit_Errors: enums.ResultStatus2: decimal Number of detected bit errors

class ResultData

Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'
- Per: float: float Packet error rate Range: 0 % to 100 %, Unit: %
- Cor_Packets_Recv: int: decimal Number of correct received packets reported by the EUT Range: 0 to 30E+3
- Ber: float: float Bit error rate Range: 0 % to 100 %
- Packets_Received: int: decimal Number of received packets detected by the R&S CMW Range: 0 to 30E+3
- Num_Invalid_Crc: int: decimal Number of packets with detected CRC error
- Num_Pattern_Err: int: decimal Number of packets with detected pattern error
- Num_Payload_Err: int: decimal Number of packets with detected payload length error
- Bit_Errors: int: decimal Number of detected bit errors

calculate() → CalculateStruct

```
# SCPI: CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:PER:TMODe:LENergy:LE1M
value: CalculateStruct = driver.rxQuality.per.tmode.lowEnergy.le1M.calculate()
```

Return all results of the signaling Rx measurement in LE test mode. Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M. .) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (..:TMODe:LENergy:LRANge..) are available.

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

fetch() → ResultData

```
# SCPI: FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:PER:TMODe:LENergy:LE1M
value: ResultData = driver.rxQuality.per.tmode.lowEnergy.le1M.fetch()
```

Return all results of the signaling Rx measurement in LE test mode. Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M. .) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (..:TMODe:LENergy:LRANge..) are available.

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

read() → ResultData

```
# SCPI: READ:BLUetooth:SIGNaling<Instance>:RXQuality:PER:TMODe:LENergy:LE1M
value: ResultData = driver.rxQuality.per.tmode.lowEnergy.le1M.read()
```

Return all results of the signaling Rx measurement in LE test mode. Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M. .) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (..:TMODe:LENergy:LRANge..) are available.

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

7.9.4.4.1.2 Le2M

SCPI Commands

```
CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:PER:TMODe:LENergy:LE2M
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:PER:TMODe:LENergy:LE2M
READ:BLUetooth:SIGNaling<Instance>:RXQuality:PER:TMODe:LENergy:LE2M
```

class Le2M

Le2M commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'
- Per: float: float Packet error rate Range: 0 % to 100 %, Unit: %
- Cor_Packets_Recv: float: decimal Number of correct received packets reported by the EUT Range: 0 to 30E+3
- Ber: float: float Bit error rate Range: 0 % to 100 %
- Packets_Received: float: decimal Number of received packets detected by the R&S CMW Range: 0 to 30E+3
- Num_Invalid_Crc: enums.ResultStatus2: decimal Number of packets with detected CRC error
- Num_Pattern_Err: enums.ResultStatus2: decimal Number of packets with detected pattern error
- Num_Payload_Err: enums.ResultStatus2: decimal Number of packets with detected payload length error
- Bit_Errors: enums.ResultStatus2: decimal Number of detected bit errors

class ResultData

Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'
- Per: float: float Packet error rate Range: 0 % to 100 %, Unit: %
- Cor_Packets_Recv: int: decimal Number of correct received packets reported by the EUT Range: 0 to 30E+3
- Ber: float: float Bit error rate Range: 0 % to 100 %
- Packets_Received: int: decimal Number of received packets detected by the R&S CMW Range: 0 to 30E+3
- Num_Invalid_Crc: int: decimal Number of packets with detected CRC error
- Num_Pattern_Err: int: decimal Number of packets with detected pattern error
- Num_Payload_Err: int: decimal Number of packets with detected payload length error
- Bit_Errors: int: decimal Number of detected bit errors

calculate() → CalculateStruct

```
# SCPI: CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:PER:TMODe:LENergy:LE2M
value: CalculateStruct = driver.rxQuality.per.tmode.lowEnergy.le2M.calculate()
```

Return all results of the signaling Rx measurement in LE test mode. Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M. .) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (..:TMODe:LENergy:LRANge..) are available.

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

fetch() → ResultData

```
# SCPI: FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:PER:TMODe:LENergy:LE2M
value: ResultData = driver.rxQuality.per.tmode.lowEnergy.le2M.fetch()
```

Return all results of the signaling Rx measurement in LE test mode. Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M. .) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (..:TMODe:LENergy:LRANge..) are available.

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

read() → ResultData

```
# SCPI: READ:BLUetooth:SIGNaling<Instance>:RXQuality:PER:TMODe:LENergy:LE2M
value: ResultData = driver.rxQuality.per.tmode.lowEnergy.le2M.read()
```

Return all results of the signaling Rx measurement in LE test mode. Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M. .) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (..:TMODe:LENergy:LRANge..) are available.

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

7.9.4.4.1.3 Lrange

SCPI Commands

```
CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:PER:TMODe:LENergy:LRANge
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:PER:TMODe:LENergy:LRANge
READ:BLUetooth:SIGNaling<Instance>:RXQuality:PER:TMODe:LENergy:LRANge
```

class Lrange

Lrange commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'
- Per: float: float Packet error rate Range: 0 % to 100 %, Unit: %
- Cor_Packets_Recv: float: decimal Number of correct received packets reported by the EUT Range: 0 to 30E+3
- Ber: float: float Bit error rate Range: 0 % to 100 %

- Packets_Received: float: decimal Number of received packets detected by the R&S CMW Range: 0 to 30E+3
- Num_Invalid_Crc: enums.ResultStatus2: decimal Number of packets with detected CRC error
- Num_Pattern_Err: enums.ResultStatus2: decimal Number of packets with detected pattern error
- Num_Payload_Err: enums.ResultStatus2: decimal Number of packets with detected payload length error
- Bit_Errors: enums.ResultStatus2: decimal Number of detected bit errors

class ResultData

Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'
- Per: float: float Packet error rate Range: 0 % to 100 %, Unit: %
- Cor_Packets_Recv: int: decimal Number of correct received packets reported by the EUT Range: 0 to 30E+3
- Ber: float: float Bit error rate Range: 0 % to 100 %
- Packets_Received: int: decimal Number of received packets detected by the R&S CMW Range: 0 to 30E+3
- Num_Invalid_Crc: int: decimal Number of packets with detected CRC error
- Num_Pattern_Err: int: decimal Number of packets with detected pattern error
- Num_Payload_Err: int: decimal Number of packets with detected payload length error
- Bit_Errors: int: decimal Number of detected bit errors

calculate() → CalculateStruct

```
# SCPI: CALCulate:BLUetooth:SIGNaling<Instance>
↪:RXQuality:PER:TMODe:LENergy:LRANge
value: CalculateStruct = driver.rxQuality.per.tmode.lowEnergy.lrange.calculate()
```

Return all results of the signaling Rx measurement in LE test mode. Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M. .) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (..:TMODe:LENergy:LRANge..) are available.

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

fetch() → ResultData

```
# SCPI: FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:PER:TMODe:LENergy:LRANge
value: ResultData = driver.rxQuality.per.tmode.lowEnergy.lrange.fetch()
```

Return all results of the signaling Rx measurement in LE test mode. Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M. .) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (..:TMODe:LENergy:LRANge..) are available.

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

read() → ResultData

```
# SCPI: READ:BLUetooth:SIGNaling<Instance>:RXQuality:PER:TMODe:LENergy:LRANge
value: ResultData = driver.rxQuality.per.tmode.lowEnergy.lrange.read()
```

Return all results of the signaling Rx measurement in LE test mode. Commands for uncoded LE 1M PHY (..:TMODe:LENergy:LE1M. .) , LE 2M PHY (..:TMODe:LENergy:LE2M..) , and LE coded PHY (..:TMODe:LENergy:LRANge..) are available.

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

7.9.5 Ber

class Ber

Ber commands group definition. 8 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.9.5.1 State

class State

State commands group definition. 2 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.9.5.1.1 All

class All

All commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.rxQuality.ber.state.all.clone()
```

Subgroups

7.9.5.1.1.1 Bedr

SCPI Commands

```
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:BER:STATe:ALL:BEDR
```

class Bedr

Bedr commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Main_State: enums.ResourceState: OFF | RDY | RUN OFF: measurement switched off, no resources allocated, no results available (when entered after STOP...) RDY: measurement has been terminated, valid results can be available RUN: measurement running (after INITiate..., READ...) , synchronization pending or adjusted, resources active or queued
- Sync_State: enums.ResourceState: PEND | ADJ | INV PEND: waiting for resource allocation, adjustment, hardware switching ('pending') ADJ: all necessary adjustments finished, measurement running ('adjusted') INV: not applicable because main_state: OFF or RDY ('invalid')
- Resource_State: enums.ResourceState: QUE | ACT | INV QUE: measurement without resources, no results available ('queued') ACT: resources allocated, acquisition of results in progress but not complete ('active') INV: not applicable because main_state: OFF or RDY ('invalid')

fetch() → FetchStruct

```
# SCPI: FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:BER:STATe:ALL[:BEDR]
value: FetchStruct = driver.rxQuality.ber.state.all.bedr.fetch()
```

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.

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

7.9.5.1.2 Bedr

SCPI Commands

```
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:BER:STATe:BEDR
```

class Bedr

Bedr commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch() → RsCmwBluetoothSig.enums.ResourceState

```
# SCPI: FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:BER:STATe[:BEDR]
value: enums.ResourceState = driver.rxQuality.ber.state.bedr.fetch()
```

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.

return meas_status: OFF | RDY | RUN OFF: measurement switched off, no resources allocated, no results available (when entered after ABORt...) RDY: measurement has been terminated, valid results can be available RUN: measurement running (after INITiate..., READ...) , synchronization pending or adjusted, resources active or queued

7.9.5.2 Bedr

SCPI Commands

```
ABORt:BLUetooth:SIGNaling<Instance>:RXQuality:BER:BEDR
INITiate:BLUetooth:SIGNaling<Instance>:RXQuality:BER:BEDR
STOP:BLUetooth:SIGNaling<Instance>:RXQuality:BER:BEDR
READ:BLUetooth:SIGNaling<Instance>:RXQuality:BER:BEDR
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:BER:BEDR
CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:BER:BEDR
```

class Bedr

Bedr commands group definition. 6 total commands, 0 Sub-groups, 6 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'
- Ber: float: float Bit error rate Range: 0 % to 100 %, Unit: %
- Per: float: float Packet error rate Range: 0 % to 100 %, Unit: %
- Bit_Errors: float: decimal Sum of received erroneous data bits Range: 0 to 18.4467440737096E+18
- Missing_Packets: float: float Difference between the number of packets sent and the number of packets received in percentage Range: 0 % to 100 %, Unit: %
- Nak: float: float Percentage of packets not acknowledged by the EUT positively Range: 0 % to 100 %, Unit: %
- Hec_Errors: float: float Percentage of packets with the bit errors in the header Range: 0 % to 100 %, Unit: %
- Crc_Errors: float: float Percentage of packets with the bit errors in the payload Range: 0 % to 100 %, Unit: %
- Wrong_Packet_Rate: float: No parameter help available
- Wrong_Payload_Rat: float: No parameter help available
- Packets_Received: float: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: decimal See 'Reliability Indicator'

- Ber: float: float Bit error rate Range: 0 % to 100 %, Unit: %
- Per: float: float Packet error rate Range: 0 % to 100 %, Unit: %
- Bit_Errors: int: decimal Sum of received erroneous data bits Range: 0 to 18.4467440737096E+18
- Missing_Packets: float: float Difference between the number of packets sent and the number of packets received in percentage Range: 0 % to 100 %, Unit: %
- Nak: float: float Percentage of packets not acknowledged by the EUT positively Range: 0 % to 100 %, Unit: %
- Hec_Errors: float: float Percentage of packets with the bit errors in the header Range: 0 % to 100 %, Unit: %
- Crc_Errors: float: float Percentage of packets with the bit errors in the payload Range: 0 % to 100 %, Unit: %
- Wrong_Packet_Rate: float: No parameter help available
- Wrong_Payload_Rat: float: No parameter help available
- Packets_Received: int: No parameter help available

abort() → None

```
# SCPI: ABORt:BLUetooth:SIGNaling<Instance>:RXQuality:BER[:BEDR]
driver.rxQuality.ber.bedr.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.

abort_with_opc() → None

```
# SCPI: ABORt:BLUetooth:SIGNaling<Instance>:RXQuality:BER[:BEDR]
driver.rxQuality.ber.bedr.abort_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.
```

(continues on next page)

(continued from previous page)

Use FETCh...STATe? to query the current measurement state.

Same as abort, but waits for the operation to complete before continuing further. Use the RsCmwBluetoothSig.utilities.opc_timeout_set() to set the timeout value.

calculate() → CalculateStruct

```
# SCPI: CALCulate:BLUetooth:SIGNaling<Instance>:RXQuality:BER[:BEDR]
value: CalculateStruct = driver.rxQuality.ber.bedr.calculate()
```

Return all results of the signaling BER 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:BLUetooth:SIGNaling<Instance>:RXQuality:BER[:BEDR]
value: ResultData = driver.rxQuality.ber.bedr.fetch()
```

Return all results of the signaling BER 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.

initiate() → None

```
# SCPI: INITiate:BLUetooth:SIGNaling<Instance>:RXQuality:BER[:BEDR]
driver.rxQuality.ber.bedr.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.

initiate_with_opc() → None

```
# SCPI: INITiate:BLUetooth:SIGNaling<Instance>:RXQuality:BER[:BEDR]
driver.rxQuality.ber.bedr.initiate_with_opc()
```

(continues on next page)

(continued from previous page)

```
    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 initiate, but waits for the operation to complete before continuing further. Use the RsCmwBluetoothSig.utilities.opc_timeout_set() to set the timeout value.

read() → ResultData

```
# SCPI: READ:BLUetooth:SIGNaling<Instance>:RXQuality:BER[:BEDR]
value: ResultData = driver.rxQuality.ber.bedr.read()
```

Return all results of the signaling BER 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.

stop() → None

```
# SCPI: STOP:BLUetooth:SIGNaling<Instance>:RXQuality:BER[:BEDR]
driver.rxQuality.ber.bedr.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() → None

```
# SCPI: STOP:BLUetooth:SIGNaling<Instance>:RXQuality:BER[:BEDR]
driver.rxQuality.ber.bedr.stop_with_opc()
```

(continues on next page)

(continued from previous page)

```
    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 RsCmwBluetoothSig.utilities.opc_timeout_set() to set the timeout value.

7.9.6 Trace

class Trace

Trace commands group definition. 8 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.9.6.1 IqCoherency

class IqCoherency

IqCoherency commands group definition. 8 total commands, 4 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.9.6.1.1 A0Reference

SCPI Commands

```
READ:BLUetooth:SIGNaling<Instance>:RXQuality:TRACe:IQCoherency:A0Reference
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:TRACe:IQCoherency:A0Reference
```

class A0Reference

A0Reference commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:BLUetooth:SIGNaling<Instance>
↪:RXQuality:TRACe:IQCoherency:A0Reference
value: List[float] = driver.rxQuality.trace.iqCoherency.a0Reference.fetch()
```

Return the trace results of reference phase deviation RPD for IQ samples coherency measured at antenna 0.

Use RsCmwBluetoothSig.reliability.last_value to read the updated reliability indicator.

return mrp: float 720 RPD results, one result per IQ sample slot.

read() → List[float]

```
# SCPI: READ:BLUetooth:SIGNaling<Instance>
↪:RXQuality:TRACe:IQCoherency:A0Reference
value: List[float] = driver.rxQuality.trace.iqCoherency.a0Reference.read()
```

Return the trace results of reference phase deviation RPD for IQ samples coherency measured at antenna 0.

Use RsCmwBluetoothSig.reliability.last_value to read the updated reliability indicator.

return mrp: float 720 RPD results, one result per IQ sample slot.

7.9.6.1.2 A1Nreference

SCPI Commands

```
READ:BLUetooth:SIGNaling<Instance>:RXQuality:TRACe:IQCoherency:A1NReference
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:TRACe:IQCoherency:A1NReference
```

class A1Nreference

A1Nreference commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:BLUetooth:SIGNaling<Instance>
↪:RXQuality:TRACe:IQCoherency:A1NReference
value: List[float] = driver.rxQuality.trace.iqCoherency.a1Nreference.fetch()
```

Return the trace results of relative phase values RP(m) for IQ samples coherency Rx measurements. Commands for mandatory second (non-reference) antenna (...:A1NReference) , and optional third and fourth non-reference antennas are available.

Use RsCmwBluetoothSig.reliability.last_value to read the updated reliability indicator.

return mrp: No help available

read() → List[float]

```
# SCPI: READ:BLUetooth:SIGNaling<Instance>
↪:RXQuality:TRACe:IQCoherency:A1NReference
value: List[float] = driver.rxQuality.trace.iqCoherency.a1Nreference.read()
```

Return the trace results of relative phase values RP(m) for IQ samples coherency Rx measurements. Commands for mandatory second (non-reference) antenna (...:A1NReference) , and optional third and fourth non-reference antennas are available.

Use RsCmwBluetoothSig.reliability.last_value to read the updated reliability indicator.

return mrp: No help available

7.9.6.1.3 A2Nreference

SCPI Commands

```
READ:BLUetooth:SIGNaling<Instance>:RXQuality:TRACe:IQCoherency:A2NReference
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:TRACe:IQCoherency:A2NReference
```

class A2Nreference

A2Nreference commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:BLUetooth:SIGNaling<Instance>
↪:RXQuality:TRACe:IQCoherency:A2NReference
value: List[float] = driver.rxQuality.trace.iqCoherency.a2Nreference.fetch()
```

Return the trace results of relative phase values RP(m) for IQ samples coherency Rx measurements. Commands for mandatory second (non-reference) antenna (...:A1NReference) , and optional third and fourth non-reference antennas are available.

Use RsCmwBluetoothSig.reliability.last_value to read the updated reliability indicator.

return mrp: No help available

read() → List[float]

```
# SCPI: READ:BLUetooth:SIGNaling<Instance>
↪:RXQuality:TRACe:IQCoherency:A2NReference
value: List[float] = driver.rxQuality.trace.iqCoherency.a2Nreference.read()
```

Return the trace results of relative phase values RP(m) for IQ samples coherency Rx measurements. Commands for mandatory second (non-reference) antenna (...:A1NReference) , and optional third and fourth non-reference antennas are available.

Use RsCmwBluetoothSig.reliability.last_value to read the updated reliability indicator.

return mrp: No help available

7.9.6.1.4 A3Nreference

SCPI Commands

```
READ:BLUetooth:SIGNaling<Instance>:RXQuality:TRACe:IQCoherency:A3NReference
FETCh:BLUetooth:SIGNaling<Instance>:RXQuality:TRACe:IQCoherency:A3NReference
```

class A3Nreference

A3Nreference commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:BLUetooth:SIGNaling<Instance>
↪:RXQuality:TRACe:IQCoherency:A3NReference
value: List[float] = driver.rxQuality.trace.iqCoherency.a3Nreference.fetch()
```

Return the trace results of relative phase values RP(m) for IQ samples coherency Rx measurements. Commands for mandatory second (non-reference) antenna (...:A1NReference) , and optional third and fourth non-reference antennas are available.

Use RsCmwBluetoothSig.reliability.last_value to read the updated reliability indicator.

return mrp: No help available

read() → List[float]

```
# SCPI: READ:BLUetooth:SIGNaling<Instance>
↪:RXQuality:TRACe:IQCoherency:A3NReference
value: List[float] = driver.rxQuality.trace.iqCoherency.a3Nreference.read()
```

Return the trace results of relative phase values RP(m) for IQ samples coherency Rx measurements. Commands for mandatory second (non-reference) antenna (...:A1NReference) , and optional third and fourth non-reference antennas are available.

Use RsCmwBluetoothSig.reliability.last_value to read the updated reliability indicator.

return mrp: No help available

7.10 Clean

class Clean

Clean commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.10.1 Elogging

SCPI Commands

```
CLEan:BLUetooth:SIGNaling<Instance>:ELOGging
```

class Elogging

Elogging commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

set() → None

```
# SCPI: CLEan:BLUetooth:SIGNaling<Instance>:ELOGging
driver.clean.elogging.set()
```

Clears the event log.

set_with_opc() → None

```
# SCPI: CLEan:BLUetooth:SIGNaling<Instance>:ELOGging
driver.clean.elogging.set_with_opc()
```

Clears the event log.

Same as set, but waits for the operation to complete before continuing further. Use the RsCmwBluetoothSig.utilities.opc_timeout_set() to set the timeout value.

INDEX

CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STA
160
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STA
141
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STA
160
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STA
141
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STA
155
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STA
141
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STA
155
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STA
139
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STA
155
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STA
139
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STA
158
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STA
139
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STA
158
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STA
137
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STA
158
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STA
137
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STA
162
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STA
137
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STA
162
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STA
145
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STA
168
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STA
146
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STA
168
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STA
146
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STA
168
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STA
152
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STA
163
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STA
152
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STA
163
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STA
147
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STA
166
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STA
147
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STA
166
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STA
150
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STA
133
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STA
150
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:DTX:STA
145
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:EATTenu
267
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:EATTenu
134
267
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:ENPower
134
129
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:FREQuer
143
260
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:FREQuer
143
260
CONFigure:BLUetooth:SIGNaling<Instance>:RFSettings:FREQuer
143
260

D

F

I

R

S