RsCMX_Signaling

Release 7.70.0.18

Rohde & Schwarz

Apr 19, 2024

CONTENTS:


5G
CMX/P

CHAPTER

ONE

REVISION HISTORY

1.1 RsCMX_Signaling

Rohde & Schwarz CMX Signaling RsCMX_Signaling instrument driver.

Basic Hello-World code:

```
from RsCMX_Signaling import *

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

Supported instruments: CMX500

The package is hosted here: https://pypi.org/project/RsCMX-Signaling/

Documentation: https://RsCMX-Signaling.readthedocs.io/

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

1.1.1 Version history

Latest release notes summary: Update for FW 7.70

Version 7.70.0

- Update for FW 7.70

Version 7.40.0

- Update for FW 7.40

Version 4.0.120

- Update of RsCMX_Signaling to FW 4.0.120 from the complete FW package 7.10.0

Version 4.0.110

- Update of RsCMX_Signaling to FW 4.0.110

Version 4.0.60

- Update of RsCMX_Signaling to FW 4.0.60

Version 4.0.10

- First released version

CHAPTER

TWO

GETTING STARTED

2.1 Introduction

RsCMX_Signaling 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 example for RsCmpx-Base and RsCmpx-Gprf:

```
"""
# GitHub examples repository path: CMXP/Python/RsCmxp_xxx_ScpiPackages

Example on how to use the python RsCmx auto-generated instrument drivers for
RsCmpx_Base and RsCmpx_Gprf (Base and GPRF) in one script with shared VISA session.
"""

from RsCMPX_Base.RsCMPX_Base import RsCMPX_Base  # install from pypi.org
from RsCMPX_Base import enums as base_enums
from RsCMPX_Base import repcap as base_repcap

from RsCMPX_Gprf.RsCMPX_Gprf import RsCMPX_Gprf  # install from pypi.org
from RsCMPX_Gprf.CustomFiles.reliability import ReliabilityEventArgs
from RsCMPX_Gprf import enums as gprf_enums
from RsCMPX_Gprf import repcap as gprf_repcaps
```

(continues on next page)

(continued from previous page)

```
# CMX Base init
cmx_base = RsCMPX_Base('TCPIP::10.112.1.116', False, True)
print(f'CMX Base IND: {cmx_base.utilities.idn_string}')
print(f'CMX Instrument options:\n{",".join(cmx_base.utilities.instrument_options)}')
cmx_base.utilities.visa_timeout = 5000  # default is 10000

# Sends OPC after each command
cmx_base.utilities.opc_query_after_write = False
# Checks for syst:err? after each command / query - default value after init is True
cmx_base.utilities.instrument_status_checking = True

# Self-test
self_test = cmx_base.utilities.self_test()
print(f'CMW self-test result: {self_test} - {"Passed" if self_test[0] == 0 else "Failed"}
↪"')
# Reference Frequency Source
cmx_base.system.reference.frequency.source_set(base_enums.SourceIntExt.INTernal)

# CMX RsCMPX_Gprf Init - reuse the session of the cmx_base, rather than creating another␣
↪one
cmx_gprf = RsCMPX_Gprf.from_existing_session(cmx_base)
cmx_gprf.utilities.visa_timeout = 5000

# Driver's Interface reliability offers a convenient way of reacting on the return value␣
↪Reliability Indicator
cmx_gprf.reliability.ExceptionOnError = True  # default is 10000


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


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

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

# Close the sessions
cmx_gprf.close()
cmx_base.close()
```

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

- Type-safe API using typing module
- You can still use the plain SCPI communication
- You can select which VISA to use or even not use any VISA at all
- Initialization of a new session is straight-forward, no need to set any other properties

- Many useful features are already implemented - reset, self-test, opc-synchronization, error checking, option checking
- Binary data blocks transfer in both directions
- Transfer of arrays of numbers in binary or ASCII format
- File transfers in both directions
- Events generation in case of error, sent data, received data, chunk data (for big files transfer)
- Multithreading session locking - you can use multiple threads talking to one instrument at the same time
- Logging feature tailored for SCPI communication - different for binary and ascii data

2.2 Installation

RsCMX_Signaling 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 RsCMX_Signaling`

Option 2 - Installing in Pycharm

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

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

Option 3 - Offline Installation

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

- Download this python script (**Save target as**): rsinstrument_offline_install.py This installs all the preconditions that the RsCMX_Signaling needs.
- Execute the script in your offline computer (supported is python 3.6 or newer)
- Download the RsCMX_Signaling package to your computer from the pypi.org: https://pypi.org/project/RsCMX_Signaling/#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\RsCMX_Signaling-7.70.0.18.tar`

2.3 Finding Available Instruments

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

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

from RsCMX_Signaling import *

# Use the instr_list string items as resource names in the RsCMX_Signaling constructor
instr_list = RsCMX_Signaling.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 RsCMX_Signaling 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 = RsCMX_Signaling.list_resources('?*', 'rs')
print(instr_list)
```

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

- Small footprint
- Superior VXI-11 and HiSLIP performance
- Integrated legacy sensors NRP-Zxx support

- Additional VXI-11 and LXI devices search
- Availability for Windows, Linux, Mac OS

2.4 Initiating Instrument Session

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

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

- Installed RsCMX_Signaling Python module Version 7.70.0 or newer from pypi.org
- Installed VISA, for example R&S Visa 5.12 or newer
"""

from RsCMX_Signaling import *

# A good practice is to assure that you have a certain minimum version installed
RsCMX_Signaling.assert_minimum_version('7.70.0')
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 = RsCMX_Signaling(resource_string_1)

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

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

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

Do not care about specialty of each session kind; RsCMX_Signaling 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 = RsCMX_Signaling('TCPIP::192.168.56.101::hislip0', id_query=True, reset=True)
```

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

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

from RsCMX_Signaling import *

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

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

(continues on next page)

(continued from previous page)

```
from RsCMX_Signaling import *

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

More `option_string` tokens are separated by comma:

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

Shared Session

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

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

from RsCMX_Signaling import *

driver1 = RsCMX_Signaling('TCPIP::192.168.56.101::INSTR', True, True)
driver2 = RsCMX_Signaling.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}')
```

(continues on next page)

(continued from previous page)

```
driver1.close()
print(f'driver1: Only now I am closed.')
```

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

2.5 Plain SCPI Communication

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

driver = RsCMX_Signaling('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 RsCMX_Signaling import *

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

# 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 RsCMX_Signaling raises an exception. Speaking of exceptions, an important feature of the RsCMX_Signaling 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 RsCMX_Signaling import *

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

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

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

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

to an appropriate value to prevent the timeout exception. Here's the snippet:

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

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

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

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

2.6 Error Checking

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

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

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

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

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

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

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

See the next chapter on how to react on errors.

2.7 Exception Handling

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

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

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

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

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

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

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

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

Tip: General rules for exception handling:

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

2.8 Transferring Files

Instrument -> PC

You definitely experienced it: you just did a perfect measurement, saved the results as a screenshot to an instrument's storage drive. Now you want to transfer it to your PC. With RsCMX_Signaling, 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 RsCMX_Signaling one-liner split into 3 lines:

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

2.9 Writing Binary Data

Writing from bytes

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

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

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

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

Writing from PC files

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

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

2.10 Transferring Big Data with Progress

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

RsCMX_Signaling 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 RsCMX_Signaling 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 = RsCMX_Signaling('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 RsCMX_Signaling does not know the size of the complete response up-front. However, if you use the same mechanism for transfer of a known data size (for example, file transfer), you get the information about the total size too, and hence you can calculate the progress as:

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

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

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

2.11 Multithreading

You are at the party, many people talking over each other. Not every person can deal with such crosstalk, neither can measurement instruments. For this reason, RsCMX_Signaling 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 RsCMX_Signaling object
"""

import threading
from RsCMX_Signaling import *
```

(continues on next page)

(continued from previous page)

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


driver = RsCMX_Signaling('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 RsCMX_Signaling objects with shared session
"""

import threading
from RsCMX_Signaling import *


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

threads = []
```

(continues on next page)

(continued from previous page)

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

Run the following example:

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

import threading
from RsCMX_Signaling import *


def execute(session: RsCMX_Signaling, 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 = RsCMX_Signaling('TCPIP::192.168.56.101::INSTR')
driver2 = RsCMX_Signaling('TCPIP::192.168.56.101::INSTR')
driver1.utilities.visa_timeout = 200
```

(continues on next page)

(continued from previous page)

```
driver2.utilities.visa_timeout = 200

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

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

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

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

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

2.12 Logging

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

What can you actually do with the logger?

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

Let us take this basic example:

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

from RsCMX_Signaling import *

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

(continues on next page)

(continued from previous page)

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

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

Console output:

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

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

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

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

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

See the full logger help *here*.

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

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

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

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

from RsCMX_Signaling import *

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

(continues on next page)

(continued from previous page)

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

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

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

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

driver.utilities.reset()

# Close the session
driver.close()

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

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

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

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

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

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

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

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

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

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

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

(continues on next page)

(continued from previous page)

```
from RsCMX_Signaling import *

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

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

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

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

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

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

Console output:

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

Notice the following:

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

CHAPTER

THREE

ENUMS

3.1 AckOrDtx

```
# Example value:
value = enums.AckOrDtx.CONTinue
# All values (2x):
CONTinue | STOP
```

3.2 Action

```
# Example value:
value = enums.Action.CONNect
# All values (2x):
CONNect | DISConnect
```

3.3 AggrLevel

```
# Example value:
value = enums.AggrLevel.N0
# All values (8x):
N0 | N1 | N2 | N3 | N4 | N5 | N6 | N8
```

3.4 Algorithm

```
# Example value:
value = enums.Algorithm.ERC1
# All values (4x):
ERC1 | ERC2 | ERC3 | ERC4
```

3.5 Alpha

```
# Example value:
value = enums.Alpha.A00
# All values (8x):
A00 | A04 | A05 | A06 | A07 | A08 | A09 | A10
```

3.6 AntennaLayout

```
# First value:
value = enums.AntennaLayout.N121
# Last value:
value = enums.AntennaLayout.TX2
# All values (14x):
N121 | N161 | N21 | N22 | N32 | N41 | N42 | N43
N44 | N61 | N62 | N81 | N82 | TX2
```

3.7 AntNoPorts

```
# Example value:
value = enums.AntNoPorts.P1
# All values (3x):
P1 | P2 | P4
```

3.8 AntNoPortsB

```
# Example value:
value = enums.AntNoPortsB.P1
# All values (4x):
P1 | P2 | P4 | P8
```

3.9 Aoa

```
# Example value:
value = enums.Aoa.AOA1
# All values (3x):
AOA1 | AOA2 | CONDucted
```

3.10 AoaB

```
# Example value:
value = enums.AoaB.AOA1
# All values (5x):
AOA1 | AOA2 | AOA3 | AOA4 | CONDucted
```

3.11 Asn1SignalMode

```
# Example value:
value = enums.Asn1SignalMode.B1
# All values (4x):
B1 | B2 | B4 | UECap
```

3.12 Assignment

```
# Example value:
value = enums.Assignment.NONE
# All values (8x):
NONE | SA0 | SA1 | SA2 | SA3 | SA4 | SA5 | SA6
```

3.13 Association

```
# Example value:
value = enums.Association.CSIRs
# All values (2x):
CSIRs | SSBBeam
```

3.14 AswitchingType

```
# Example value:
value = enums.AswitchingType.T1R1
# All values (6x):
T1R1 | T1R2 | T1R4 | T2R2 | T2R4 | T4R4
```

3.15 AuthProcedure

```
# Example value:
value = enums.AuthProcedure.EAKA
# All values (2x):
EAKA | FAKA
```

3.16 AutoMode

```
# Example value:
value = enums.AutoMode.AUTO
# All values (3x):
AUTO | OFF | ON
```

3.17 BandwidthCommon

```
# Example value:
value = enums.BandwidthCommon.BW0
# All values (8x):
BW0 | BW1 | BW2 | BW3 | BW4 | BW5 | BW6 | BW7
```

3.18 BandwidthDedicated

```
# Example value:
value = enums.BandwidthDedicated.BW0
# All values (4x):
BW0 | BW1 | BW2 | BW3
```

3.19 BandwidthHoping

```
# Example value:
value = enums.BandwidthHoping.HBW0
# All values (4x):
HBW0 | HBW1 | HBW2 | HBW3
```

3.20 BeamConfigMode

```
# Example value:
value = enums.BeamConfigMode.ALL
# All values (3x):
ALL | AUTO | UDEFined
```

3.21 BeamNoPorts

```
# Example value:
value = enums.BeamNoPorts.NONE
# All values (5x):
NONE | P1 | P2 | P4 | P8
```

3.22 BeamsTrigger

```
# Example value:
value = enums.BeamsTrigger.ACTive
# All values (5x):
ACTive | ID0 | ID1 | ID2 | ID3
```

3.23 BlerState

```
# Example value:
value = enums.BlerState.FAIL
# All values (3x):
FAIL | PASS | PENDing
```

3.24 BurstType

```
# Example value:
value = enums.BurstType.FBURst
# All values (2x):
FBURst | RBURst
```

3.25 BwidthTotal

```
# First value:
value = enums.BwidthTotal.B100
# Last value:
value = enums.BwidthTotal.B700
# All values (13x):
B100 | B150 | B200 | B250 | B300 | B350 | B400 | B450
B500 | B550 | B600 | B650 | B700
```

3.26 BwpSwitchingMode

```
# Example value:
value = enums.BwpSwitchingMode.DYNamic
# All values (2x):
DYNamic | STATic
```

3.27 BwSelection

```
# Example value:
value = enums.BwSelection.ALL
# All values (2x):
ALL | RB52
```

3.28 CcrntisEnd

```
# Example value:
value = enums.CcrntisEnd.ASF
# All values (5x):
ASF | BLSF | F2SF | LSF | SASF
```

3.29 CellDeployment

```
# Example value:
value = enums.CellDeployment.REAL
# All values (2x):
REAL | VIRTual
```

3.30 CellPucchFormatPy

```
# Example value:
value = enums.CellPucchFormatPy.F0
# All values (5x):
F0 | F1 | F2 | F3 | F4
```

3.31 CellsToMeasure

```
# Example value:
value = enums.CellsToMeasure.ALL
# All values (6x):
ALL | LAA | LLAA | LTE | NRADio | OFF
```

3.32 CellsTypeToMeasure

```
# Example value:
value = enums.CellsTypeToMeasure.CELLs
# All values (2x):
CELLs | CGRoup
```

3.33 CellType

```
# Example value:
value = enums.CellType.LTE
# All values (2x):
LTE | NR
```

3.34 Choice

```
# Example value:
value = enums.Choice.CODebook
# All values (3x):
CODebook | NCODebook | SINGle
```

3.35 CipherAlgorithm

```
# First value:
value = enums.CipherAlgorithm.EA0
# Last value:
value = enums.CipherAlgorithm.HIGHest
# All values (9x):
EA0 | EA1 | EA2 | EA3 | EA4 | EA5 | EA6 | EA7
HIGHest
```

3.36 Class

```
# Example value:
value = enums.Class.C0
# All values (4x):
C0 | C1 | C2 | C3
```

3.37 CodebookSubset

```
# Example value:
value = enums.CodebookSubset.AUTO
# All values (4x):
AUTO | FPNC | NC | PNC
```

3.38 Coding

```
# Example value:
value = enums.Coding.EIGHt
# All values (3x):
EIGHt | GSM | UCS2
```

3.39 CodingGroup

```
# Example value:
value = enums.CodingGroup.G7
# All values (3x):
G7 | G7L | U2L
```

3.40 ConfigMode

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

3.41 ConfigType

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

3.42 ConfigTypeB

```
# Example value:
value = enums.ConfigTypeB.T1
# All values (3x):
T1 | T2 | T3
```

3.43 Control

```
# Example value:
value = enums.Control.CLOop
# All values (5x):
CLOop | KEEP | MAX | MIN | PATTern
```

3.44 CoreNetwork

```
# Example value:
value = enums.CoreNetwork.EPS
# All values (2x):
EPS | FG
```

3.45 Counter

```
# First value:
value = enums.Counter.N1
# Last value:
value = enums.Counter.N8
# All values (9x):
N1 | N10 | N2 | N20 | N3 | N4 | N5 | N6
N8
```

3.46 DataFlow

```
# Example value:
value = enums.DataFlow.MCG
# All values (4x):
MCG | MCGSplit | SCG | SCGSplit
```

3.47 DciFormat

```
# First value:
value = enums.DciFormat.D0
# Last value:
value = enums.DciFormat.D2D
# All values (10x):
D0 | D1 | D1A | D1B | D1C | D2 | D2A | D2B
D2C | D2D
```

3.48 DciFormatB

```
# Example value:
value = enums.DciFormatB.D10
# All values (2x):
D10 | D11
```

3.49 DciFormatC

```
# Example value:
value = enums.DciFormatC.D00
# All values (2x):
D00 | D01
```

3.50 DcMode

```
# Example value:
value = enums.DcMode.ENDC
# All values (5x):
ENDC | LTE | NR | NRDC | OFF
```

3.51 DensityPreset

```
# Example value:
value = enums.DensityPreset.NPResent
# All values (2x):
NPResent | PRESent
```

3.52 DiagBaseband

```
# Example value:
value = enums.DiagBaseband.BBCombining
# All values (2x):
BBCombining | MRFPerf
```

3.53 DiagCellSignal

```
# Example value:
value = enums.DiagCellSignal.COMBining
# All values (4x):
COMBining | OTA | OTASep | SEParation
```

3.54 DisplayMode

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

3.55 DlIqDataStreams

```
# Example value:
value = enums.DlIqDataStreams.S1
# All values (4x):
S1 | S2 | S4 | S8
```

3.56 DlUlBandwidth

```
# First value:
value = enums.DlUlBandwidth.B005
# Last value:
value = enums.DlUlBandwidth.M90
# All values (30x):
B005 | B010 | B015 | B020 | B025 | B030 | B040 | B050
B060 | B070 | B080 | B090 | B100 | B200 | B400 | M10
M100 | M15 | M20 | M200 | M25 | M30 | M40 | M400
M5 | M50 | M60 | M70 | M80 | M90
```

3.57 DlUlLocation

```
# Example value:
value = enums.DlUlLocation.HIGH
# All values (4x):
HIGH | LOW | MID | USER
```

3.58 DuplexModeB

```
# Example value:
value = enums.DuplexModeB.FDD
# All values (3x):
FDD | SDL | TDD
```

3.59 EdRxMode

```
# Example value:
value = enums.EdRxMode.UERequested
# All values (2x):
UERequested | USER
```

3.60 EnableCqi

```
# Example value:
value = enums.EnableCqi.APERiodic
# All values (4x):
APERiodic | OFF | PERiodic | SPERsistant
```

3.61 EpreRatio

```
# Example value:
value = enums.EpreRatio.R0
# All values (2x):
R0 | R1
```

3.62 EpsRejectCause

```
# First value:
value = enums.EpsRejectCause.C002
# Last value:
value = enums.EpsRejectCause.C113
# All values (72x):
C002 | C003 | C005 | C006 | C007 | C008 | C009 | C010
C011 | C012 | C013 | C014 | C015 | C016 | C017 | C018
C019 | C020 | C021 | C022 | C023 | C024 | C025 | C026
C027 | C028 | C029 | C030 | C031 | C032 | C033 | C034
C035 | C036 | C037 | C038 | C039 | C040 | C041 | C042
C043 | C044 | C045 | C046 | C047 | C049 | C050 | C051
C052 | C053 | C054 | C055 | C056 | C057 | C058 | C059
C060 | C061 | C065 | C066 | C078 | C081 | C095 | C096
C097 | C098 | C099 | C100 | C101 | C111 | C112 | C113
```

3.63 EpsRejectProcedure

```
# Example value:
value = enums.EpsRejectProcedure.ATTR
# All values (5x):
ATTR | BEAR | NOR | PDNR | TAUR
```

3.64 EsmCause

```
# First value:
value = enums.EsmCause.C100
# Last value:
value = enums.EsmCause.C99
# All values (45x):
C100 | C101 | C111 | C112 | C113 | C16 | C26 | C27
C28 | C29 | C30 | C31 | C32 | C33 | C34 | C35
C36 | C37 | C38 | C39 | C41 | C42 | C43 | C44
C45 | C46 | C47 | C49 | C50 | C51 | C52 | C53
C54 | C55 | C56 | C59 | C60 | C65 | C66 | C81
C95 | C96 | C97 | C98 | C99
```

3.65 FadingMode

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

3.66 FadingProfile

```
# First value:
value = enums.FadingProfile.CTES
# Last value:
value = enums.FadingProfile.UMIL
# All values (110x):
CTES | EP5A | EP5H | EP5L | EP5M | EPAE | EPHE | EPLE
EPME | ET1A | ET1H | ET1L | ET1M | ET3A | ET3H | ET3L
ET3M | ET7A | ET7H | ET7L | ET7M | ETA3 | ETAE | ETHA
ETHE | ETLA | ETLE | ETMA | ETME | EV5A | EV5H | EV5L
EV5M | EV7A | EV7H | EV7L | EV7M | EVAE | EVHE | EVLE
EVME | HST | HST2 | HSTS | INHL | INHN | MBSF | NH
NHD | NHDF | NHDI | NHE | NHG | NHH | NHI | NHS
NHSF | NHSI | NNAC | NNC5 | NONE | RMAL | RMAN | SMAL
SMAN | TAAA | TAAM | TAAN | TAHA | TAHM | TAHN | TALA
TALM | TALN | TAMA | TAMM | TAMN | TBAC | TBHC | TBLC
TBMC | TCAD | TCAO | TCHD | TCHO | TCLD | TCLO | TCMD
TCMO | TDAA | TDAM | TDAN | TDHA | TDHM | TDHN | TDLA
TDLM | TDLN | TDMA | TDMM | TDMN | UMA3 | UMAA | UMAL
UMAN | UMI1 | UMI2 | UMI3 | UMIA | UMIL
```

3.67 FgsRejectCause

```
# First value:
value = enums.FgsRejectCause.C003
# Last value:
value = enums.FgsRejectCause.C111
# All values (77x):
C003 | C005 | C006 | C007 | C008 | C009 | C010 | C011
C012 | C013 | C015 | C020 | C021 | C022 | C023 | C024
C026 | C027 | C028 | C029 | C031 | C032 | C033 | C034
C035 | C036 | C037 | C038 | C039 | C041 | C042 | C043
C044 | C045 | C046 | C047 | C050 | C051 | C054 | C057
C058 | C059 | C061 | C062 | C065 | C067 | C068 | C069
C070 | C071 | C072 | C073 | C074 | C075 | C076 | C077
C078 | C079 | C080 | C081 | C082 | C083 | C084 | C085
C086 | C090 | C091 | C092 | C093 | C095 | C096 | C097
C098 | C099 | C100 | C101 | C111
```

3.68 FgsRejectProcedure

```
# Example value:
value = enums.FgsRejectProcedure.AUTR
# All values (4x):
AUTR | NOR | PDUR | REGR
```

3.69 FilterCoeff

```
# First value:
value = enums.FilterCoeff.FC0
# Last value:
value = enums.FilterCoeff.FC9
# All values (15x):
FC0 | FC1 | FC11 | FC13 | FC15 | FC17 | FC19 | FC2
FC3 | FC4 | FC5 | FC6 | FC7 | FC8 | FC9
```

3.70 FlowControl

```
# Example value:
value = enums.FlowControl.GUARanteed
# All values (2x):
GUARanteed | NGUaranteed
```

3.71 FollowCqi

```
# Example value:
value = enums.FollowCqi.DISabled
# All values (5x):
DISabled | MSB | UEBSubband | UEPSubband | WB
```

3.72 FollowPmi

```
# Example value:
value = enums.FollowPmi.DISabled
# All values (4x):
DISabled | SB | WB | WBEXplicit
```

3.73 FollowRi

```
# Example value:
value = enums.FollowRi.DISabled
# All values (3x):
DISabled | ENABled | RETX
```

3.74 FollowType

```
# Example value:
value = enums.FollowType.DISabled
# All values (2x):
DISabled | ENABled
```

3.75 FormatCqi

```
# Example value:
value = enums.FormatCqi.SB
# All values (2x):
SB | WB
```

3.76 Frame

```
# Example value:
value = enums.Frame.T16
# All values (5x):
T16 | T1T | T2 | T4 | T8
```

3.77 FramesOffset

```
# Example value:
value = enums.FramesOffset.T16
# All values (8x):
T16 | T1T | T2 | T2T | T32 | T4 | T4T | T8
```

3.78 FrequencyRange

```
# Example value:
value = enums.FrequencyRange.FR1
# All values (2x):
FR1 | FR2
```

3.79 FtpMode

```
# Example value:
value = enums.FtpMode.AUTO
# All values (5x):
AUTO | FULL | MOD1 | MOD2 | OFF
```

3.80 GroupLanguage

```
# Example value:
value = enums.GroupLanguage.G7L
# All values (2x):
G7L | U2L
```

3.81 IdentityType

```
# Example value:
value = enums.IdentityType.GCI
# All values (4x):
GCI | GLI | IMSI | NAI
```

3.82 IgnorePrachMode

```
# Example value:
value = enums.IgnorePrachMode.IALLways
# All values (3x):
IALLways | IXTimes | RALLways
```

3.83 IndicationMode

```
# Example value:
value = enums.IndicationMode.AUTO
# All values (4x):
AUTO | OATime | WAT1 | WAT2
```

3.84 Info

```
# Example value:
value = enums.Info.ALL
# All values (3x):
ALL | DL | UL
```

3.85 InitialSfAlloc

```
# Example value:
value = enums.InitialSfAlloc.S0
# All values (2x):
S0 | S7
```

3.86 IntegrityAlgorithm

```
# First value:
value = enums.IntegrityAlgorithm.HIGHest
# Last value:
value = enums.IntegrityAlgorithm.IA7
# All values (9x):
HIGHest | IA0 | IA1 | IA2 | IA3 | IA4 | IA5 | IA6
IA7
```

3.87 Ira

```
# Example value:
value = enums.Ira.E2
# All values (4x):
E2 | E3 | E4 | E8
```

3.88 ItRateUnit

```
# First value:
value = enums.ItRateUnit.G1
# Last value:
value = enums.ItRateUnit.T64
# All values (25x):
G1 | G16 | G256 | G4 | G64 | K1 | K16 | K256
K4 | K64 | M1 | M16 | M256 | M4 | M64 | P1
P16 | P256 | P4 | P64 | T1 | T16 | T256 | T4
T64
```

3.89 Ktc

```
# Example value:
value = enums.Ktc.N2
# All values (2x):
N2 | N4
```

3.90 LanguageB

```
# First value:
value = enums.LanguageB.DANish
# Last value:
value = enums.LanguageB.UNSPecified
# All values (16x):
DANish | DUTCh | ENGLish | FINNish | FRENch | GERMan | GREek | HUNGarian
ITALian | NORWegian | POLish | PORTuguese | SPANish | SWEDish | TURKish | UNSPecified
```

3.91 Level

```
# Example value:
value = enums.Level.AL1
# All values (5x):
AL1 | AL16 | AL2 | AL4 | AL8
```

3.92 LimitStatus

```
# Example value:
value = enums.LimitStatus.APRogress
# All values (4x):
APRogress | DPRogress | OFF | ON
```

3.93 Location

```
# Example value:
value = enums.Location.HIGH
# All values (3x):
HIGH | LOW | MID
```

3.94 LogFileState

```
# Example value:
value = enums.LogFileState.AVAilable
# All values (3x):
AVAilable | CREating | NERunning
```

3.95 LogLevel

```
# Example value:
value = enums.LogLevel.BRIef
# All values (3x):
BRIef | NONE | VERBose
```

3.96 LogType

```
# Example value:
value = enums.LogType.DISable
# All values (4x):
DISable | FULL | HEADer | PAYLoad
```

3.97 LowHigh

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

3.98 LteMimoScheme

```
# Example value:
value = enums.LteMimoScheme.M2N
# All values (4x):
M2N | M4N | S1N | UDEFined
```

3.99 Mapping

```
# Example value:
value = enums.Mapping.A
# All values (2x):
A | B
```

3.100 MappingI

```
# Example value:
value = enums.MappingI.INT
# All values (2x):
INT | NINT
```

3.101 MaxLength

```
# Example value:
value = enums.MaxLength.L1
# All values (2x):
L1 | L2
```

3.102 MaxPorts

```
# Example value:
value = enums.MaxPorts.N1
# All values (2x):
N1 | N2
```

3.103 McsBehavior

```
# Example value:
value = enums.McsBehavior.AUTO
# All values (4x):
AUTO | REPeat | REPLace | SUBStitute
```

3.104 McsMode

```
# Example value:
value = enums.McsMode.FIXed
# All values (3x):
FIXed | MAX | MMO
```

3.105 McsTable

```
# Example value:
value = enums.McsTable.Q1K
# All values (3x):
Q1K | Q256 | Q64
```

3.106 McsTableB

```
# Example value:
value = enums.McsTableB.L64
# All values (3x):
L64 | Q256 | Q64
```

3.107 McsTableC

```
# Example value:
value = enums.McsTableC.AUTO
# All values (3x):
AUTO | P521 | UDEFined
```

3.108 McsTableD

```
# Example value:
value = enums.McsTableD.Q16
# All values (3x):
Q16 | Q256 | Q64
```

3.109 Mimo

```
# Example value:
value = enums.Mimo.M22
# All values (4x):
M22 | M33 | M44 | SISO
```

3.110 MimoB

```
# Example value:
value = enums.MimoB.M22
# All values (2x):
M22 | SISO
```

3.111 Mode

```
# Example value:
value = enums.Mode.BINDex
# All values (3x):
BINDex | CSIRs | SSBBeam
```

3.112 ModeB

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

3.113 ModeBfollow

```
# Example value:
value = enums.ModeBfollow.AUTO
# All values (3x):
AUTO | BLOCk | OFF
```

3.114 ModeC

```
# Example value:
value = enums.ModeC.AUTO
# All values (3x):
AUTO | NOTC | USER
```

3.115 ModeD

```
# Example value:
value = enums.ModeD.MAX
# All values (3x):
MAX | MIN | UDEFined
```

3.116 ModeE

```
# First value:
value = enums.ModeE.CPRI
# Last value:
value = enums.ModeE.UDEFined
# All values (9x):
CPRI | CQI | CRI | FIXed | PMI | PRI | RI | SPS
UDEFined
```

3.117 ModeFrecovery

```
# Example value:
value = enums.ModeFrecovery.AUTO
# All values (3x):
AUTO | OFF | UDEFined
```

3.118 ModeFrecoveryB

```
# Example value:
value = enums.ModeFrecoveryB.HADamard
# All values (5x):
HADamard | IDENtity | OFF | TGPP | UDEFined
```

3.119 ModeRvs

```
# Example value:
value = enums.ModeRvs.AUTO
# All values (3x):
AUTO | S101 | UDEFined
```

3.120 ModeS

```
# Example value:
value = enums.ModeS.FIXed
# All values (4x):
FIXed | SPS | SRBSr | UDEFined
```

3.121 ModeSrs

```
# Example value:
value = enums.ModeSrs.A508
# All values (4x):
A508 | A521 | OFF | UDEFined
```

3.122 ModeTrs

```
# Example value:
value = enums.ModeTrs.DEF
# All values (3x):
DEF | OFF | UDEF
```

3.123 ModeUeCapability

```
# Example value:
value = enums.ModeUeCapability.AUTO
# All values (3x):
AUTO | SKIP | UDEFined
```

3.124 ModeUeScheduling

```
# Example value:
value = enums.ModeUeScheduling.BO
# All values (7x):
BO | CPRI | CQI | FIXed | PRI | SPS | UDEFined
```

3.125 Modulation

```
# Example value:
value = enums.Modulation.BPSK
# All values (7x):
BPSK | P2BPsk | Q1024 | Q16 | Q256 | Q64 | QPSK
```

3.126 ModulationB

```
# Example value:
value = enums.ModulationB.BPSK
# All values (7x):
BPSK | P2BPsk | Q16 | Q1K | Q256 | Q64 | QPSK
```

3.127 ModulationOrder

```
# Example value:
value = enums.ModulationOrder.Q16
# All values (5x):
Q16 | Q1K | Q256 | Q64 | QPSK
```

3.128 ModulationRetr

```
# Example value:
value = enums.ModulationRetr.AUTO
# All values (8x):
AUTO | BPSK | HPBP | Q16 | Q1K | Q256 | Q64 | QPSK
```

3.129 MtxPosition

```
# Example value:
value = enums.MtxPosition.P0
# All values (4x):
P0 | P1 | P2 | P3
```

3.130 NameType

```
# Example value:
value = enums.NameType.GUI
# All values (2x):
GUI | RESource
```

3.131 NcellsToMeasure

```
# Example value:
value = enums.NcellsToMeasure.ALL
# All values (5x):
ALL | IAFRequency | IFRequency | IRAT | OFF
```

3.132 NcellType

```
# Example value:
value = enums.NcellType.IAFRequency
# All values (3x):
IAFRequency | IFRequency | IRAT
```

3.133 NcoherentTpmi

```
# Example value:
value = enums.NcoherentTpmi.FPARtial
# All values (2x):
FPARtial | NCOHerent
```

3.134 NeighborCellType

```
# Example value:
value = enums.NeighborCellType.CNETwork
# All values (3x):
CNETwork | NCList | SIB
```

3.135 NoSymbols

```
# Example value:
value = enums.NoSymbols.S1
# All values (4x):
S1 | S2 | S3 | S4
```

3.136 NoSymbolsN

```
# Example value:
value = enums.NoSymbolsN.N1
# All values (3x):
N1 | N2 | N4
```

3.137 OfdmSymbols

```
# Example value:
value = enums.OfdmSymbols.ALL
# All values (7x):
ALL | S10 | S11 | S12 | S3 | S6 | S9
```

3.138 OnDurationTimer

```
# First value:
value = enums.OnDurationTimer.M1
# Last value:
value = enums.OnDurationTimer.M9D
# All values (55x):
M1 | M10 | M100 | M10D | M11D | M12D | M13D | M14D
M15D | M16D | M17D | M18D | M19D | M1D | M1K0 | M1K2
M1K6 | M2 | M20 | M200 | M20D | M21D | M22D | M23D
M24D | M25D | M26D | M27D | M28D | M29D | M2D | M3
M30 | M300 | M30D | M31D | M3D | M4 | M40 | M400
M4D | M5 | M50 | M500 | M5D | M6 | M60 | M600
M6D | M7D | M8 | M80 | M800 | M8D | M9D
```

3.139 PagingCycle

```
# Example value:
value = enums.PagingCycle.P128
# All values (4x):
P128 | P256 | P32 | P64
```

3.140 PannelType

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

3.141 Pattern

```
# Example value:
value = enums.Pattern.D1
# All values (4x):
D1 | KEEP | U1 | U3
```

3.142 PcellNr

```
# Example value:
value = enums.PcellNr.B050
# All values (4x):
B050 | B100 | B200 | B400
```

3.143 PdcchFormat

```
# Example value:
value = enums.PdcchFormat.N1
# All values (5x):
N1 | N2 | N4 | N8 | NAV
```

3.144 PdcchFormatB

```
# Example value:
value = enums.PdcchFormatB.N1
# All values (4x):
N1 | N2 | N4 | N8
```

3.145 PduState

```
# Example value:
value = enums.PduState.ACTive
# All values (6x):
ACTive | AIP | AUIP | DIP | INACtive | MIP
```

3.146 Periodicity

```
# First value:
value = enums.Periodicity.P0P5
# Last value:
value = enums.Periodicity.P5
# All values (10x):
P0P5 | P0P6 | P1 | P10 | P1P2 | P2 | P2P5 | P3
P4 | P5
```

3.147 PeriodicityB

```
# Example value:
value = enums.PeriodicityB.P10
# All values (6x):
P10 | P160 | P20 | P40 | P5 | P80
```

3.148 PeriodicityCqiReport

```
# First value:
value = enums.PeriodicityCqiReport.P10
# Last value:
value = enums.PeriodicityCqiReport.UDEFined
# All values (11x):
P10 | P16 | P160 | P20 | P320 | P4 | P40 | P5
P8 | P80 | UDEFined
```

3.149 PeriodicityRsrc

```
# First value:
value = enums.PeriodicityRsrc.P10
# Last value:
value = enums.PeriodicityRsrc.P80
# All values (13x):
P10 | P16 | P160 | P20 | P32 | P320 | P4 | P40
P5 | P64 | P640 | P8 | P80
```

3.150 Ports

```
# Example value:
value = enums.Ports.P1
# All values (8x):
P1 | P12 | P16 | P2 | P24 | P32 | P4 | P8
```

3.151 Power

```
# First value:
value = enums.Power.P100
# Last value:
value = enums.Power.P98
# All values (16x):
P100 | P102 | P104 | P106 | P108 | P110 | P112 | P114
P116 | P118 | P120 | P90 | P92 | P94 | P96 | P98
```

3.152 PowerScaling

```
# Example value:
value = enums.PowerScaling.TGPP
# All values (2x):
TGPP | TOPTimized
```

3.153 PowerStatus

```
# Example value:
value = enums.PowerStatus.IRANge
# All values (4x):
IRANge | ODRiven | OFF | UDRiven
```

3.154 Predefined3Gpp

```
# First value:
value = enums.Predefined3Gpp.M1
# Last value:
value = enums.Predefined3Gpp.M9
# All values (38x):
M1 | M10 | M11 | M11A | M11B | M12 | M12A | M12B
M13 | M14 | M15 | M16 | M17 | M18 | M19 | M1A
M1B | M2 | M20 | M21 | M22 | M23 | M24 | M25
M26 | M27 | M28 | M29 | M2A | M3 | M3A | M4
M4A | M5 | M6 | M7 | M8 | M9
```

3.155 PreferredNetw

```
# Example value:
value = enums.PreferredNetw.AUTO
# All values (4x):
AUTO | EPS | FG | NONE
```

3.156 ProhibitTimer

```
# First value:
value = enums.ProhibitTimer.INF
# Last value:
value = enums.ProhibitTimer.S90
# All values (24x):
INF | S0 | S0D4 | S0D5 | S0D8 | S1 | S10 | S12
S120 | S1D6 | S2 | S20 | S3 | S30 | S300 | S4
S5 | S6 | S60 | S600 | S7 | S8 | S9 | S90
```

3.157 Prtype

```
# Example value:
value = enums.Prtype.OFF
# All values (3x):
OFF | PRTA | PRTB
```

3.158 PsOrder

```
# Example value:
value = enums.PsOrder.RROBin
# All values (2x):
RROBin | SBOund
```

3.159 PtrsPower

```
# Example value:
value = enums.PtrsPower.P00
# All values (2x):
P00 | P01
```

3.160 PwrRampingStepA

```
# Example value:
value = enums.PwrRampingStepA.S0
# All values (4x):
S0 | S2 | S4 | S6
```

3.161 PwrRampingStepB

```
# Example value:
value = enums.PwrRampingStepB.S0
# All values (4x):
S0 | S2 | S3 | S4
```

3.162 Qi

```
# First value:
value = enums.Qi.Q1
# Last value:
value = enums.Qi.Q9
# All values (21x):
Q1 | Q2 | Q3 | Q4 | Q5 | Q6 | Q65 | Q66
Q67 | Q69 | Q7 | Q70 | Q75 | Q79 | Q8 | Q80
Q82 | Q83 | Q84 | Q85 | Q9
```

3.163 Quantity

```
# First value:
value = enums.Quantity.OFF
# Last value:
value = enums.Quantity.Q9
# All values (10x):
OFF | Q1 | Q2 | Q3 | Q4 | Q5 | Q6 | Q7
Q8 | Q9
```

3.164 RangeChoice

```
# Example value:
value = enums.RangeChoice.HASYmmetric
# All values (5x):
HASYmmetric | HIGH | LOW | MID | UDEFined
```

3.165 RedCapId

```
# Example value:
value = enums.RedCapId.MSG3
# All values (4x):
MSG3 | PRACh | UECap | UNSPecified
```

3.166 RegState

```
# First value:
value = enums.RegState.DREG
# Last value:
value = enums.RegState.RIP
# All values (9x):
DREG | DRIP | FREG | FRIP | NFReg | NREG | NRIP | REG
RIP
```

3.167 RegStateB

```
# First value:
value = enums.RegStateB.CREG
# Last value:
value = enums.RegStateB.REG
# All values (9x):
CREG | CRIP | DREG | DRIP | EREG | ERIP | LREG | LRIP
REG
```

3.168 Repeat

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

3.169 Repetitions

```
# Example value:
value = enums.Repetitions.N12
# All values (8x):
N12 | N16 | N2 | N3 | N4 | N7 | N8 | OFF
```

3.170 ReportCqi

```
# Example value:
value = enums.ReportCqi.OFF
# All values (3x):
OFF | SB | WB
```

3.171 ReportInterval

```
# First value:
value = enums.ReportInterval.I1
# Last value:
value = enums.ReportInterval.I9
# All values (14x):
I1 | I10 | I11 | I12 | I13 | I14 | I2 | I3
I4 | I5 | I6 | I7 | I8 | I9
```

3.172 ReportMode

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

3.173 ReportType

```
# Example value:
value = enums.ReportType.APERiodic
# All values (3x):
APERiodic | OFF | PERiodic
```

3.174 ResourceAllocationType

```
# Example value:
value = enums.ResourceAllocationType.DSWich
# All values (3x):
DSWich | T0 | T1
```

3.175 ResourceId

```
# Example value:
value = enums.ResourceId.R1
# All values (2x):
R1 | R2
```

3.176 ResourceOffset

```
# Example value:
value = enums.ResourceOffset.NPResent
# All values (4x):
NPResent | OF01 | OF10 | OF11
```

3.177 ReTxBehavior

```
# Example value:
value = enums.ReTxBehavior.FLUSh
# All values (3x):
FLUSh | NAPPlicable | RETain
```

3.178 ReTxBehaviorB

```
# Example value:
value = enums.ReTxBehaviorB.CONTinue
# All values (4x):
CONTinue | SDTX | SNDMimo | STOP
```

3.179 RgbSize

```
# Example value:
value = enums.RgbSize.CON1
# All values (2x):
CON1 | CON2
```

3.180 Riv

```
# Example value:
value = enums.Riv.NADaptive
# All values (2x):
NADaptive | NEW
```

3.181 RlcMode

```
# Example value:
value = enums.RlcMode.ACK
# All values (4x):
ACK | UACK | UADL | UAUL
```

3.182 RnauTimer

```
# First value:
value = enums.RnauTimer.M10
# Last value:
value = enums.RnauTimer.OFF
# All values (9x):
M10 | M120 | M20 | M30 | M360 | M5 | M60 | M720
OFF
```

3.183 Routing

```
# Example value:
value = enums.Routing.DUT
# All values (2x):
DUT | FIXed
```

3.184 RpPattern

```
# Example value:
value = enums.RpPattern.A
# All values (3x):
A | B | C
```

3.185 RrcState

```
# Example value:
value = enums.RrcState.CONNected
# All values (3x):
CONNected | IDLE | INACtive
```

3.186 RsrcPower

```
# Example value:
value = enums.RsrcPower.M3DB
# All values (6x):
M3DB | M6DB | M9DB | P3DB | P6DB | ZERO
```

3.187 Schema

```
# Example value:
value = enums.Schema.CODebook
# All values (2x):
CODebook | NCODebook
```

3.188 SecurityAlgorithm

```
# Example value:
value = enums.SecurityAlgorithm.AES
# All values (4x):
AES | OFF | SNOW | ZUC
```

3.189 SecurityAlgorithmB

```
# First value:
value = enums.SecurityAlgorithmB.EEA0
# Last value:
value = enums.SecurityAlgorithmB.HIGHest
# All values (9x):
EEA0 | EEA1 | EEA2 | EEA3 | EEA4 | EEA5 | EEA6 | EEA7
HIGHest
```

3.190 SecurityAlgorithmC

```
# First value:
value = enums.SecurityAlgorithmC.EIA0
# Last value:
value = enums.SecurityAlgorithmC.HIGHest
# All values (9x):
EIA0 | EIA1 | EIA2 | EIA3 | EIA4 | EIA5 | EIA6 | EIA7
HIGHest
```

3.191 Severity

```
# Example value:
value = enums.Severity.ERRor
# All values (3x):
ERRor | INFO | WARNing
```

3.192 SpecialPattern

```
# First value:
value = enums.SpecialPattern.P0
# Last value:
value = enums.SpecialPattern.PAV2
# All values (12x):
P0 | P1 | P2 | P3 | P4 | P5 | P6 | P7
P8 | P9 | PAV1 | PAV2
```

3.193 Spreset

```
# Example value:
value = enums.Spreset.S1
# All values (3x):
S1 | S2 | S3
```

3.194 SpsPadding

```
# Example value:
value = enums.SpsPadding.ALLZero
# All values (2x):
ALLZero | NOPadding
```

3.195 SpsPeriodicity

```
# First value:
value = enums.SpsPeriodicity.S1
# Last value:
value = enums.SpsPeriodicity.SYM7
# All values (25x):
S1 | S10 | S128 | S16 | S160 | S1K | S1K2 | S2
S20 | S256 | S2K | S32 | S320 | S4 | S40 | S5
S512 | S5K | S64 | S640 | S8 | S80 | SYM2 | SYM6
SYM7
```

3.196 SpsPosition

```
# Example value:
value = enums.SpsPosition.POS0
# All values (4x):
POS0 | POS1 | POS2 | POS3
```

3.197 SrcType

```
# Example value:
value = enums.SrcType.PUCC
# All values (3x):
PUCC | PUPU | PUSC
```

3.198 State

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

3.199 StateCnetwork

```
# First value:
value = enums.StateCnetwork.CREating
# Last value:
value = enums.StateCnetwork.TESTing
# All values (10x):
CREating | DELeting | ERRor | EXHausted | IDLE | NAV | RUNNing | STARting
STOPping | TESTing
```

3.200 StatePwrControl

```
# Example value:
value = enums.StatePwrControl.RDY
# All values (2x):
RDY | RUN
```

3.201 StateTest

```
# Example value:
value = enums.StateTest.ERRor
# All values (2x):
ERRor | SUCCess
```

3.202 StopCondition

```
# Example value:
value = enums.StopCondition.CONFidence
# All values (3x):
CONFidence | SAMPles | TIME
```

3.203 SubCarrSpacing

```
# Example value:
value = enums.SubCarrSpacing.A15
# All values (5x):
A15 | B30 | C30 | D120 | E240
```

3.204 Subframe

```
# First value:
value = enums.Subframe.SC0
# Last value:
value = enums.Subframe.SC9
# All values (16x):
SC0 | SC1 | SC10 | SC11 | SC12 | SC13 | SC14 | SC15
SC2 | SC3 | SC4 | SC5 | SC6 | SC7 | SC8 | SC9
```

3.205 SymbolPair

```
# First value:
value = enums.SymbolPair.S04
# Last value:
value = enums.SymbolPair.S913
# All values (10x):
S04 | S15 | S26 | S37 | S48 | S59 | S610 | S711
S812 | S913
```

3.206 TadvPeriodicity

```
# Example value:
value = enums.TadvPeriodicity.CONTinuous
# All values (3x):
CONTinuous | OFF | SINGleshot
```

3.207 Target

```
# Example value:
value = enums.Target.ALL
# All values (5x):
ALL | CELL | LTE | NRADio | TOPology
```

3.208 TargetCellScg

```
# Example value:
value = enums.TargetCellScg.RELease
# All values (1x):
RELease
```

3.209 Tdd

```
# Example value:
value = enums.Tdd.CP1
# All values (3x):
CP1 | CP2 | SEParation
```

3.210 TdType

```
# Example value:
value = enums.TdType.APERiodic
# All values (3x):
APERiodic | PERiodic | PERSistent
```

3.211 TestFunction

```
# Example value:
value = enums.TestFunction.RX
# All values (3x):
RX | RXTX | TX
```

3.212 TestLoopState

```
# Example value:
value = enums.TestLoopState.CLOSe
# All values (2x):
CLOSe | OPEN
```

3.213 TimeOffset

```
# Example value:
value = enums.TimeOffset.T0
# All values (7x):
T0 | T10 | T15 | T20 | T40 | T5 | T80
```

3.214 TimerUnit

```
# Example value:
value = enums.TimerUnit.DEACtivated
# All values (8x):
DEACtivated | H1 | H10 | H320 | M1 | M10 | S2 | S30
```

3.215 TimerUnitB

```
# Example value:
value = enums.TimerUnitB.DEACtivated
# All values (4x):
DEACtivated | M1 | M6 | S2
```

3.216 Tmode

```
# First value:
value = enums.Tmode.TM1
# Last value:
value = enums.Tmode.TM9
# All values (10x):
TM1 | TM10 | TM2 | TM3 | TM4 | TM5 | TM6 | TM7
TM8 | TM9
```

3.217 TpcDirection

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

3.218 TpControl

```
# Example value:
value = enums.TpControl.CLOop
# All values (6x):
CLOop | KEEP | MAX | MIN | PATTern | RPTolerance
```

3.219 Tpmi

```
# Example value:
value = enums.Tpmi.T0
# All values (6x):
T0 | T1 | T2 | T3 | T4 | T5
```

3.220 TpTimeDens

```
# Example value:
value = enums.TpTimeDens.D2
# All values (2x):
D2 | NPResent
```

3.221 TrsPeriodicity

```
# Example value:
value = enums.TrsPeriodicity.P10
# All values (4x):
P10 | P20 | P40 | P80
```

3.222 TxRxSeparation

```
# Example value:
value = enums.TxRxSeparation.DEFault
# All values (2x):
DEFault | UDEFined
```

3.223 Type

```
# Example value:
value = enums.Type.DCMC
# All values (2x):
DCMC | GDC
```

3.224 TypeB

```
# Example value:
value = enums.TypeB.UDEFined
# All values (1x):
UDEFined
```

3.225 TypeDlUl

```
# Example value:
value = enums.TypeDlUl.RMC
# All values (2x):
RMC | UDEFined
```

3.226 UecState

```
# Example value:
value = enums.UecState.CESTablish
# All values (7x):
CESTablish | CREestablish | CRELease | HANDover | OK | PAGing | SCGFailure
```

3.227 UeScFactor

```
# Example value:
value = enums.UeScFactor.N2
# All values (4x):
N2 | N4 | N8 | OFF
```

3.228 UeType

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

3.229 UlBandwidth

```
# First value:
value = enums.UlBandwidth.B014
# Last value:
value = enums.UlBandwidth.M5
# All values (12x):
B014 | B030 | B050 | B100 | B150 | B200 | M10 | M15
M1K4 | M20 | M3 | M5
```

3.230 UlEnable

```
# Example value:
value = enums.UlEnable.OFF
# All values (3x):
OFF | ON | SRS
```

3.231 UlIndication

```
# Example value:
value = enums.UlIndication.AOFF
# All values (3x):
AOFF | AON | AUTO
```

3.232 UlMaxDutyCyle

```
# Example value:
value = enums.UlMaxDutyCyle.D80
# All values (7x):
D80 | D82 | D85 | D87 | D89 | OFF | ON
```

3.233 VcCalibQuantity

```
# Example value:
value = enums.VcCalibQuantity.CRITical
# All values (3x):
CRITical | GOOD | INSufficient
```

3.234 Version

```
# Example value:
value = enums.Version.AUTO
# All values (5x):
AUTO | RV0 | RV1 | RV2 | RV3
```

3.235 VoiceHandling

```
# Example value:
value = enums.VoiceHandling.EFHandover
# All values (4x):
EFHandover | EFRedirect | UECap | VONR
```

3.236 Waveform

```
# Example value:
value = enums.Waveform.CP
# All values (2x):
CP | DTFS
```

3.237 WusMode

```
# Example value:
value = enums.WusMode.RATio
# All values (2x):
RATio | UDEFined
```

CHAPTER

FOUR

REPCAPS

4.1 BwParts

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

4.2 Cword

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

4.3 MeasInstance

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

4.4 Nnum

```
# First value:
value = repcap.Nnum.Nr310
# Values (2x):
Nr310 | Nr311
```

4.5 Pattern

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

4.6 QamOrder

```
# First value:
value = repcap.QamOrder.Order64
# Values (2x):
Order64 | Order256
```

4.7 Tnum

```
# First value:
value = repcap.Tnum.Nr300
# Range:
Nr300 .. Nr319
# All values (5x):
Nr300 | Nr301 | Nr310 | Nr311 | Nr319
```

CHAPTER

FIVE

EXAMPLES

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

```
"""
# GitHub examples repository path: CMXP/Python/RsCmxp_xxx_ScpiPackages

Example on how to use the python RsCmx auto-generated instrument drivers for
RsCmpx_Base and RsCmpx_Gprf (Base and GPRF) in one script with shared VISA session.
"""

from RsCMPX_Base.RsCMPX_Base import RsCMPX_Base  # install from pypi.org
from RsCMPX_Base import enums as base_enums
from RsCMPX_Base import repcap as base_repcap

from RsCMPX_Gprf.RsCMPX_Gprf import RsCMPX_Gprf  # install from pypi.org
from RsCMPX_Gprf.CustomFiles.reliability import ReliabilityEventArgs
from RsCMPX_Gprf import enums as gprf_enums
from RsCMPX_Gprf import repcap as gprf_repcaps


# CMX Base init
cmx_base = RsCMPX_Base('TCPIP::10.112.1.116', False, True)
print(f'CMX Base IND: {cmx_base.utilities.idn_string}')
print(f'CMX Instrument options:\n{",".join(cmx_base.utilities.instrument_options)}')
cmx_base.utilities.visa_timeout = 5000  # default is 10000

# Sends OPC after each command
cmx_base.utilities.opc_query_after_write = False
# Checks for syst:err? after each command / query - default value after init is True
cmx_base.utilities.instrument_status_checking = True

# Self-test
self_test = cmx_base.utilities.self_test()
print(f'CMW self-test result: {self_test} - {"Passed" if self_test[0] == 0 else "Failed"}
↪"')
# Reference Frequency Source
cmx_base.system.reference.frequency.source_set(base_enums.SourceIntExt.INTernal)

# CMX RsCMPX_Gprf Init - reuse the session of the cmx_base, rather than creating another␣
↪one
cmx_gprf = RsCMPX_Gprf.from_existing_session(cmx_base)
```

(continues on next page)

(continued from previous page)

```
cmx_gprf.utilities.visa_timeout = 5000

# Driver's Interface reliability offers a convenient way of reacting on the return value␣
↪Reliability Indicator
cmx_gprf.reliability.ExceptionOnError = True  # default is 10000


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


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

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

# Close the sessions
cmx_gprf.close()
cmx_base.close()
```

CHAPTER

SIX

RSCMX_SIGNALING API STRUCTURE

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

1155 total commands, 14 Subgroups, 0 group commands

Initializes new RsCMX_Signaling session.

Parameter options tokens examples:

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

- `ScpiQuotes = double'. - for SCPI commands, you can define how strings are quoted. With single or double quotes. Possible values: single | double | {char}. Default: ``single`
- `LoggingMode = On` - Sets the logging status right from the start. Default: `Off`
- `LoggingName = 'MyDevice'` - Sets the name to represent the session in the log entries. Default: `'resource_name'`
- `LogToGlobalTarget = True` - Sets the logging target to the class-property previously set with RsCMX_Signaling.set_global_logging_target() Default: `False`
- `LoggingToConsole = True` - Immediately starts logging to the console. Default: False
- `LoggingToUdp = True` - Immediately starts logging to the UDP port. Default: False
- `LoggingUdpPort = 49200` - UDP port to log to. Default: 49200

Parameters

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

static assert_minimum_version(*min_version: str*) → None

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

classmethod clear_global_logging_relative_timestamp() → None

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

close() → None

Closes the active RsCMX_Signaling session.

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

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

Parameters

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

classmethod get_global_logging_relative_timestamp() → datetime

Returns global common relative timestamp for log entries.

classmethod get_global_logging_target()

Returns global common target stream.

get_session_handle() → object

Returns the underlying session handle.

get_total_execution_time() → timedelta

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

get_total_time() → timedelta

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

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

Finds all the resources defined by the expression

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

Parameters

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

reset_time_statistics() → None

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

restore_all_repcaps_to_default() → None

Sets all the Group and Global repcaps to their initial values

classmethod set_global_logging_relative_timestamp(*timestamp: datetime*) → None

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

classmethod set_global_logging_relative_timestamp_now() → None

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

classmethod set_global_logging_target(*target*) → None

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

Subgroups

6.1 Add

class AddCls

Add commands group definition. 23 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.1.1 Signaling

class SignalingCls

Signaling commands group definition. 23 total commands, 5 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.1.1.1 Eps

class EpsCls

Eps commands group definition. 3 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.1.1.1.1 UeCapability

class UeCapabilityCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.add.signaling.eps.ueCapability.clone()
```

Subgroups

6.1.1.1.1.1 Eutra

SCPI Command :

```
ADD:SIGNaling:EPS:UECapability:EUTRa:BANDs
```

class EutraCls

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

set_bands(*fbi: List[int]*) → None

```
# SCPI: ADD:SIGNaling:EPS:UECapability:EUTRa:BANDs
driver.add.signaling.eps.ueCapability.eutra.set_bands(fbi = [1, 2, 3])
```

Adds entries to the list of requested frequency bands for the container type 'UE-EUTRA-Capability', for EPS tracking areas.

param fbi
Comma-separated list of LTE frequency band indicators

6.1.1.1.2 Mrdc

class MrdcCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.add.signaling.eps.ueCapability.mrdc.clone()
```

Subgroups

6.1.1.1.3 Bands

SCPI Command :

```
ADD:SIGNaling:EPS:UECapability:MRDC:BANDs
```

class BandsCls

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

set(*rat: List[CellType]*, *fbi: List[int]*) → None

```
# SCPI: ADD:SIGNaling:EPS:UECapability:MRDC:BANDs
driver.add.signaling.eps.ueCapability.mrdc.bands.set(rat = [CellType.LTE,␣
↪CellType.NR], fbi = [1, 2, 3])
```

Adds entries to the list of requested frequency bands for the container type 'UE-MRDC-Capability', for EPS tracking areas. The bands are defined as pairs of values: {<Rat>, <Fbi>}1, {<Rat>, <Fbi>}2, ...

param rat
Type of the band: LTE band or NR band.

param fbi
Frequency band indicator

6.1.1.1.1.4 Nradio

SCPI Command :

```
ADD:SIGNaling:EPS:UECapability:NRADio:BANDs
```

class NradioCls

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

set_bands(*fbi: List[int]*) → None

```
# SCPI: ADD:SIGNaling:EPS:UECapability:NRADio:BANDs
driver.add.signaling.eps.ueCapability.nradio.set_bands(fbi = [1, 2, 3])
```

Adds entries to the list of requested frequency bands for the container type 'UE-NR-Capability', for EPS tracking areas.

param fbi
Comma-separated list of NR frequency band indicators

6.1.1.2 Fgs

class FgsCls

Fgs commands group definition. 3 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.1.1.2.1 UeCapability

class UeCapabilityCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.add.signaling.fgs.ueCapability.clone()
```

Subgroups

6.1.1.2.1.1 Eutra

SCPI Command :

```
ADD:SIGNaling:FGS:UECapability:EUTRa:BANDs
```

class EutraCls

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

set_bands(*fbi: List[int]*) → None

```
# SCPI: ADD:SIGNaling:FGS:UECapability:EUTRa:BANDs
driver.add.signaling.fgs.ueCapability.eutra.set_bands(fbi = [1, 2, 3])
```

Adds entries to the list of requested frequency bands for the container type 'UE-EUTRA-Capability', for 5GS tracking areas.

param fbi
Comma-separated list of LTE frequency band indicators

6.1.1.2.1.2 Mrdc

class MrdcCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.add.signaling.fgs.ueCapability.mrdc.clone()
```

Subgroups

6.1.1.2.1.3 Bands

SCPI Command :

```
ADD:SIGNaling:FGS:UECapability:MRDC:BANDs
```

class BandsCls

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

set(*rat: List[CellType]*, *fbi: List[int]*) → None

```
# SCPI: ADD:SIGNaling:FGS:UECapability:MRDC:BANDs
driver.add.signaling.fgs.ueCapability.mrdc.bands.set(rat = [CellType.LTE,␣
↪CellType.NR], fbi = [1, 2, 3])
```

Adds entries to the list of requested frequency bands for the container type 'UE-MRDC-Capability', for 5GS tracking areas. The bands are defined as pairs of values: {<Rat>, <Fbi>}1, {<Rat>, <Fbi>}2, ...

param rat
Type of the band: LTE band or NR band.

param fbi
Frequency band indicator

6.1.1.2.1.4 Nradio

SCPI Command :

```
ADD:SIGNaling:FGS:UECapability:NRADio:BANDs
```

class NradioCls

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

set_bands(*fbi: List[int]*) → None

```
# SCPI: ADD:SIGNaling:FGS:UECapability:NRADio:BANDs
driver.add.signaling.fgs.ueCapability.nradio.set_bands(fbi = [1, 2, 3])
```

Adds entries to the list of requested frequency bands for the container type 'UE-NR-Capability', for 5GS tracking areas.

param fbi
Comma-separated list of NR frequency band indicators

6.1.1.3 Lte

class LteCls

Lte commands group definition. 4 total commands, 3 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.1.1.3.1 Ca

class CaCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.add.signaling.lte.ca.clone()
```

Subgroups

6.1.1.3.1.1 Scell

SCPI Command :

```
ADD:SIGNaling:LTE:CA:SCELl
```

class ScellCls

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

set(*cell_group_name: str*, *cell_name: List[str]*, *activation: List[bool] = None*, *ul_enable: List[bool] = None*) → None

```
# SCPI: ADD:SIGNaling:LTE:CA:SCELl
driver.add.signaling.lte.ca.scell.set(cell_group_name = 'abc', cell_name = [
↪'abc1', 'abc2', 'abc3'], activation = [True, False, True], ul_enable = [True,␣
↪False, True])
```

Adds one or more existing LTE cells to an existing cell group, with the role SCell.

param cell_group_name
No help available

param cell_name
No help available

param activation
ON: automatic MAC activation (default) OFF: manual MAC activation via separate command

param ul_enable
Enables the UL (UL carrier aggregation) .

6.1.1.3.2 Cell

class CellCls

Cell commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.add.signaling.lte.cell.clone()
```

Subgroups

6.1.1.3.2.1 Harq

class HarqCls

Harq commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.add.signaling.lte.cell.harq.clone()
```

Subgroups

6.1.1.3.2.2 Downlink

SCPI Command :

```
ADD:SIGNaling:LTE:CELL:HARQ:DL:RETX
```

class DownlinkCls

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

class ReTxStruct

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

- Cell_Name: str: No parameter help available
- Count: int: Optional setting parameter. Number of entries to be added.
- Riv: List[enums.Riv]: Optional setting parameter. RIV non-adaptive, new TX RIV
- Tb_1: List[int]: Optional setting parameter. MCS value for first transport block
- Tb_2: List[int]: Optional setting parameter. MCS value for second transport block
- Behavior: List[enums.ReTxBehavior]: Optional setting parameter. Behavior for transport block size changes. Not applicable, flush HARQ buffer, retain HARQ buffer.

set_re_tx(*value: ReTxStruct*) → None

```
# SCPI: ADD:SIGNaling:LTE:CELL:HARQ:DL:RETX
structure = driver.add.signaling.lte.cell.harq.downlink.ReTxStruct()
structure.Cell_Name: str = 'abc'
structure.Count: int = 1
structure.Riv: List[enums.Riv] = [Riv.NADaptive, Riv.NEW]
structure.Tb_1: List[int] = [1, 2, 3]
structure.Tb_2: List[int] = [1, 2, 3]
structure.Behavior: List[enums.ReTxBehavior] = [ReTxBehavior.FLUSh,␣
↪ReTxBehavior.RETain]
driver.add.signaling.lte.cell.harq.downlink.set_re_tx(value = structure)
```

Adds entries to the end of the retransmission configuration.

param value
see the help for ReTxStruct structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.add.signaling.lte.cell.harq.downlink.clone()
```

Subgroups

6.1.1.3.2.3 RvSequence

SCPI Command :

```
ADD:SIGNaling:LTE:CELL:HARQ:DL:RVSequence
```

class RvSequenceCls

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

set(*cell_name: str*, *count: int = None*, *tb_1: List[Version] = None*, *tb_2: List[Version] = None*, *qam_64: List[Version] = None*) → None

```
# SCPI: ADD:SIGNaling:LTE:CELL:HARQ:DL:RVSequence
driver.add.signaling.lte.cell.harq.downlink.rvSequence.set(cell_name = 'abc',␣
↪count = 1, tb_1 = [Version.AUTO, Version.RV3], tb_2 = [Version.AUTO, Version.
↪RV3], qam_64 = [Version.AUTO, Version.RV3])
```

Adds entries to the end of the RV sequences.

param cell_name
No help available

param count
Number of entries to be added.

param tb_1
RV sequence for the first transport block, for QPSK and 16QAM.

param tb_2
RV sequence for the second transport block, for QPSK and 16QAM.

param qam_64
RV sequence for 64QAM and 256QAM, first and second transport block.

6.1.1.3.3 Ncell

SCPI Command :

```
ADD:SIGNaling:LTE:NCELl
```

class NcellCls

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

set(*cell_name: str*, *ncell_name: str*) → None

```
# SCPI: ADD:SIGNaling:LTE:NCELl
driver.add.signaling.lte.ncell.set(cell_name = 'abc', ncell_name = 'abc')
```

Adds a cell to the SIB neighbor cell list of an LTE or NR cell. You can define a separate neighbor cell list for each cell. The list applies only if that cell is the serving cell.

param cell_name
Name of the cell for which the neighbor is added (serving cell) .

param ncell_name
Name of the neighbor cell.

6.1.1.4 Nradio

class NradioCls

Nradio commands group definition. 11 total commands, 3 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.1.1.4.1 Ca

class CaCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.add.signaling.nradio.ca.clone()
```

Subgroups

6.1.1.4.1.1 Scell

SCPI Command :

```
ADD:SIGNaling:NRADio:CA:SCELl
```

class ScellCls

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

set(*cell_group_name: str*, *cell_name: List[str]*, *activation: List[bool] = None*, *ul_enable: List[UlEnable] = None*) → None

```
# SCPI: ADD:SIGNaling:NRADio:CA:SCELl
driver.add.signaling.nradio.ca.scell.set(cell_group_name = 'abc', cell_name = [
↪'abc1', 'abc2', 'abc3'], activation = [True, False, True], ul_enable =␣
↪[UlEnable.OFF, UlEnable.SRS])
```

Adds one or more existing NR cells to an existing cell group, with the role SCell.

param cell_group_name
No help available

param cell_name
No help available

param activation
ON: automatic MAC activation (default) OFF: manual MAC activation via separate command

param ul_enable
OFF: The SCell has no uplink. ON: The SCell has an uplink with PUSCH. SRS: PUSCH-less SCell with SRS, for SRS carrier switching.

6.1.1.4.2 Cell

class CellCls

Cell commands group definition. 9 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.add.signaling.nradio.cell.clone()
```

Subgroups

6.1.1.4.2.1 Bwp<BwParts>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.add.signaling.nradio.cell.bwp.repcap_bwParts_get()
driver.add.signaling.nradio.cell.bwp.repcap_bwParts_set(repcap.BwParts.Nr1)
```

SCPI Command :

```
ADD:SIGNaling:NRADio:CELL:BWP
```

class BwpCls

Bwp commands group definition. 5 total commands, 3 Subgroups, 1 group commands Repeated Capability: BwParts, default value after init: BwParts.Nr1

set_value(*cell_name: str*) → None

```
# SCPI: ADD:SIGNaling:NRADio:CELL:BWP
driver.add.signaling.nradio.cell.bwp.set_value(cell_name = 'abc')
```

Adds a bandwidth part (BWP) to the cell. The initial BWP is always available. Additional BWPs are numbered 1 to n.

param cell_name
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.add.signaling.nradio.cell.bwp.clone()
```

Subgroups

6.1.1.4.2.2 Csi

class CsiCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.add.signaling.nradio.cell.bwp.csi.clone()
```

Subgroups

6.1.1.4.2.3 Trs

SCPI Command :

```
ADD:SIGNaling:NRADio:CELL:BWP<bwp_id>:CSI:TRS
```

class TrsCls

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

set(*cell_name: str*, *bwParts=BwParts.Default*) → None

```
# SCPI: ADD:SIGNaling:NRADio:CELL:BWP<bwp_id>:CSI:TRS
driver.add.signaling.nradio.cell.bwp.csi.trs.set(cell_name = 'abc', bwParts =␣
↪repcap.BwParts.Default)
```

Adds a TRS configuration for BWP <bb>.

param cell_name
No help available

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

6.1.1.4.2.4 Harq

class HarqCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.add.signaling.nradio.cell.bwp.harq.clone()
```

Subgroups

6.1.1.4.2.5 Downlink

class DownlinkCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.add.signaling.nradio.cell.bwp.harq.downlink.clone()
```

Subgroups

6.1.1.4.2.6 User

class UserCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.add.signaling.nradio.cell.bwp.harq.downlink.user.clone()
```

Subgroups

6.1.1.4.2.7 Retransm

SCPI Command :

```
ADD:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:DL:USER:RETRansm
```

class RetransmCls

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

set(*cell_name: str*, *bwParts=BwParts.Default*) → None

```
# SCPI: ADD:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:DL:USER:RETRansm
driver.add.signaling.nradio.cell.bwp.harq.downlink.user.retransm.set(cell_name␣
↪= 'abc', bwParts = repcap.BwParts.Default)
```

Adds a retransmission to the retransmission configuration for user-defined DL HARQ, for BWP <bb>.

param cell_name
No help available

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

6.1.1.4.2.8 Uplink

class UplinkCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.add.signaling.nradio.cell.bwp.harq.uplink.clone()
```

Subgroups

6.1.1.4.2.9 User

class UserCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.add.signaling.nradio.cell.bwp.harq.uplink.user.clone()
```

Subgroups

6.1.1.4.2.10 Retransm

SCPI Command :

```
ADD:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:UL:USER:RETRansm
```

class RetransmCls

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

set(*cell_name: str*, *bwParts=BwParts.Default*) → None

```
# SCPI: ADD:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:UL:USER:RETRansm
driver.add.signaling.nradio.cell.bwp.harq.uplink.user.retransm.set(cell_name =
↪'abc', bwParts = repcap.BwParts.Default)
```

Adds a retransmission to the retransmission configuration for user-defined UL HARQ, for BWP <bb>.

param cell_name
No help available

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

6.1.1.4.2.11 Srs

class SrsCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.add.signaling.nradio.cell.bwp.srs.clone()
```

Subgroups

6.1.1.4.2.12 CnCodebook

class CnCodebookCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.add.signaling.nradio.cell.bwp.srs.cnCodebook.clone()
```

Subgroups

6.1.1.4.2.13 Resource

SCPI Command :

```
ADD:SIGNaling:NRADio:CELL:BWP<bwp_id>:SRS:CNCodebook:RESource
```

class ResourceCls

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

set(*cell_name: str*, *bwParts=BwParts.Default*) → None

```
# SCPI: ADD:SIGNaling:NRADio:CELL:BWP<bwp_id>:SRS:CNCodebook:RESource
driver.add.signaling.nradio.cell.bwp.srs.cnCodebook.resource.set(cell_name =
↪'abc', bwParts = repcap.BwParts.Default)
```

Adds an SRS resource to the resource set for periodic SRS, for BWP <bb>.

param cell_name
No help available

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

6.1.1.4.2.14 Csi

SCPI Command :

```
ADD:SIGNaling:NRADio:CELL:CSI:TRS
```

class CsiCls

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

set_trs(*cell_name: str*) → None

```
# SCPI: ADD:SIGNaling:NRADio:CELL:CSI:TRS
driver.add.signaling.nradio.cell.csi.set_trs(cell_name = 'abc')
```

Adds a TRS configuration for the initial BWP.

param cell_name
No help available

6.1.1.4.2.15 Harq

class HarqCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.add.signaling.nradio.cell.harq.clone()
```

Subgroups

6.1.1.4.2.16 Downlink

class DownlinkCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.add.signaling.nradio.cell.harq.downlink.clone()
```

Subgroups

6.1.1.4.2.17 User

SCPI Command :

```
ADD:SIGNaling:NRADio:CELL:HARQ:DL:USER:RETRansm
```

class UserCls

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

set_retransm(*cell_name: str*) → None

```
# SCPI: ADD:SIGNaling:NRADio:CELL:HARQ:DL:USER:RETRansm
driver.add.signaling.nradio.cell.harq.downlink.user.set_retransm(cell_name =
↪'abc')
```

Adds a retransmission to the retransmission configuration for user-defined DL HARQ, for the initial BWP.

param cell_name
No help available

6.1.1.4.2.18 Uplink

class UplinkCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.add.signaling.nradio.cell.harq.uplink.clone()
```

Subgroups

6.1.1.4.2.19 User

SCPI Command :

```
ADD:SIGNaling:NRADio:CELL:HARQ:UL:USER:RETRansm
```

class UserCls

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

set_retransm(*cell_name: str*) → None

```
# SCPI: ADD:SIGNaling:NRADio:CELL:HARQ:UL:USER:RETRansm
driver.add.signaling.nradio.cell.harq.uplink.user.set_retransm(cell_name = 'abc
↪')
```

Adds a retransmission to the retransmission configuration for user-defined UL HARQ, for the initial BWP.

param cell_name
No help available

6.1.1.4.2.20 Srs

class SrsCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.add.signaling.nradio.cell.srs.clone()
```

Subgroups

6.1.1.4.2.21 CnCodebook

SCPI Command :

```
ADD:SIGNaling:NRADio:CELL:SRS:CNCodebook:RESource
```

class CnCodebookCls

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

set_resource(*cell_name: str*) → None

```
# SCPI: ADD:SIGNaling:NRADio:CELL:SRS:CNCodebook:RESource
driver.add.signaling.nradio.cell.srs.cnCodebook.set_resource(cell_name = 'abc')
```

Adds an SRS resource to the resource set for periodic SRS, for the initial BWP.

param cell_name
No help available

6.1.1.4.3 Ncell

SCPI Command :

```
ADD:SIGNaling:NRADio:NCELl
```

class NcellCls

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

set(*cell_name: str*, *ncell_name: str*) → None

```
# SCPI: ADD:SIGNaling:NRADio:NCELl
driver.add.signaling.nradio.ncell.set(cell_name = 'abc', ncell_name = 'abc')
```

Adds a cell to the SIB neighbor cell list of an LTE or NR cell. You can define a separate neighbor cell list for each cell. The list applies only if that cell is the serving cell.

param cell_name
Name of the cell for which the neighbor is added (serving cell) .

param ncell_name
Name of the neighbor cell.

6.1.1.5 Topology

class TopologyCls

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

Cloning the Group

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

Subgroups

6.1.1.5.1 Eps

SCPI Command :

```
ADD:SIGNaling:TOPology:EPS
```

class EpsCls

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

set(*name_ta_eps: str*, *name_cell: str*) → None

```
# SCPI: ADD:SIGNaling:TOPology:EPS
driver.add.signaling.topology.eps.set(name_ta_eps = 'abc', name_cell = 'abc')
```

Associates an existing cell with an EPS tracking area.

param name_ta_eps
No help available

param name_cell
No help available

6.1.1.5.2 Fgs

SCPI Command :

```
ADD:SIGNaling:TOPology:FGS
```

class FgsCls

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

set(*name_ta_5_g: str*, *name_cell: str*) → None

```
# SCPI: ADD:SIGNaling:TOPology:FGS
driver.add.signaling.topology.fgs.set(name_ta_5_g = 'abc', name_cell = 'abc')
```

Associates an existing cell with a 5GS tracking area.

param name_ta_5_g
No help available

param name_cell
No help available

6.2 Catalog

class CatalogCls

Catalog commands group definition. 34 total commands, 5 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1 Lte

class LteCls

Lte commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.1 Measurement<MeasInstance>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.catalog.lte.measurement.repcap_measInstance_get()
driver.catalog.lte.measurement.repcap_measInstance_set(repcap.MeasInstance.Nr1)
```

class MeasurementCls

Measurement commands group definition. 2 total commands, 1 Subgroups, 0 group commands Repeated Capability: MeasInstance, default value after init: MeasInstance.Nr1

Cloning the Group

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

Subgroups

6.2.1.1.1 Network

class NetworkCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.catalog.lte.measurement.network.clone()
```

Subgroups

6.2.1.1.1.1 Cell

class CellCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.catalog.lte.measurement.network.cell.clone()
```

Subgroups

6.2.1.1.1.2 Uplinks

SCPI Command :

```
CATalog:LTE:MEASurement<Instance>:NETWork:CELL:UPLinks
```

class UplinksCls

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

get(*cell_name: str*, *measInstance=MeasInstance.Default*) → List[int]

```
# SCPI: CATalog:LTE:MEASurement<Instance>:NETWork:CELL:UPLinks
value: List[int] = driver.catalog.lte.measurement.network.cell.uplinks.get(cell_
↪name = 'abc', measInstance = repcap.MeasInstance.Default)
```

No command help available

param cell_name
No help available

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

return
available_ul: No help available

6.2.1.1.1.3 Cells

SCPI Command :

```
CATalog:LTE:MEASurement<Instance>:NETWork:CELLs
```

class CellsCls

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

get(*measInstance=MeasInstance.Default*) → List[str]

```
# SCPI: CATalog:LTE:MEASurement<Instance>:NETWork:CELLs
value: List[str] = driver.catalog.lte.measurement.network.cells.
↪get(measInstance = repcap.MeasInstance.Default)
```

No command help available

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

return
cell_name: No help available

6.2.2 NrMmw

class NrMmwCls

NrMmw commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.2.1 Measurement<MeasInstance>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.catalog.nrMmw.measurement.repcap_measInstance_get()
driver.catalog.nrMmw.measurement.repcap_measInstance_set(repcap.MeasInstance.Nr1)
```

class MeasurementCls

Measurement commands group definition. 2 total commands, 1 Subgroups, 0 group commands Repeated Capability: MeasInstance, default value after init: MeasInstance.Nr1

Cloning the Group

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

Subgroups

6.2.2.1.1 Network

class NetworkCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.catalog.nrMmw.measurement.network.clone()
```

Subgroups

6.2.2.1.1.1 Cell

class CellCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.catalog.nrMmw.measurement.network.cell.clone()
```

Subgroups

6.2.2.1.1.2 Uplinks

SCPI Command :

```
CATalog:NRMMw:MEASurement<Instance>:NETWork:CELL:UPLinks
```

class UplinksCls

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

get(*cell_name: str*, *measInstance=MeasInstance.Default*) → List[int]

```
# SCPI: CATalog:NRMMw:MEASurement<Instance>:NETWork:CELL:UPLinks
value: List[int] = driver.catalog.nrMmw.measurement.network.cell.uplinks.
↪get(cell_name = 'abc', measInstance = repcap.MeasInstance.Default)
```

No command help available

param cell_name
No help available

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

return
available_ul: No help available

6.2.2.1.1.3 Cells

SCPI Command :

```
CATalog:NRMMw:MEASurement<Instance>:NETWork:CELLs
```

class CellsCls

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

get(*measInstance=MeasInstance.Default*) → List[str]

```
# SCPI: CATalog:NRMMw:MEASurement<Instance>:NETWork:CELLs
value: List[str] = driver.catalog.nrMmw.measurement.network.cells.
↪get(measInstance = repcap.MeasInstance.Default)
```

No command help available

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

return
cell_name: No help available

6.2.3 NrSub

class NrSubCls

NrSub commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.3.1 Measurement<MeasInstance>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.catalog.nrSub.measurement.repcap_measInstance_get()
driver.catalog.nrSub.measurement.repcap_measInstance_set(repcap.MeasInstance.Nr1)
```

class MeasurementCls

Measurement commands group definition. 2 total commands, 1 Subgroups, 0 group commands Repeated Capability: MeasInstance, default value after init: MeasInstance.Nr1

Cloning the Group

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

Subgroups

6.2.3.1.1 Network

class NetworkCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.catalog.nrSub.measurement.network.clone()
```

Subgroups

6.2.3.1.1.1 Cell

class CellCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.catalog.nrSub.measurement.network.cell.clone()
```

Subgroups

6.2.3.1.1.2 Uplinks

SCPI Command :

```
CATalog:NRSub:MEASurement<Instance>:NETWork:CELL:UPLinks
```

class UplinksCls

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

get(*cell_name: str*, *measInstance=MeasInstance.Default*) → List[int]

```
# SCPI: CATalog:NRSub:MEASurement<Instance>:NETWork:CELL:UPLinks
value: List[int] = driver.catalog.nrSub.measurement.network.cell.uplinks.
↪get(cell_name = 'abc', measInstance = repcap.MeasInstance.Default)
```

No command help available

param cell_name
No help available

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

return
available_ul: No help available

6.2.3.1.1.3 Cells

SCPI Command :

```
CATalog:NRSub:MEASurement<Instance>:NETWork:CELLs
```

class CellsCls

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

get(*measInstance=MeasInstance.Default*) → List[str]

```
# SCPI: CATalog:NRSub:MEASurement<Instance>:NETWork:CELLs
value: List[str] = driver.catalog.nrSub.measurement.network.cells.
↪get(measInstance = repcap.MeasInstance.Default)
```

No command help available

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

return
cell_name: No help available

6.2.4 Signaling

SCPI Commands :

```
CATalog:SIGNaling:UE
CATalog:SIGNaling:RFCHannel
```

class SignalingCls

Signaling commands group definition. 26 total commands, 6 Subgroups, 2 group commands

get_rf_channel() → List[str]

```
# SCPI: CATalog:SIGNaling:RFCHannel
value: List[str] = driver.catalog.signaling.get_rf_channel()
```

No command help available

return
cell_name: No help available

get_ue() → List[str]

```
# SCPI: CATalog:SIGNaling:UE
value: List[str] = driver.catalog.signaling.get_ue()
```

No command help available

return
ue_name: No help available

Cloning the Group

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

Subgroups

6.2.4.1 Eps

class EpsCls

Eps commands group definition. 3 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.4.1.1 UeCapability

class UeCapabilityCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.catalog.signaling.eps.ueCapability.clone()
```

Subgroups

6.2.4.1.1.1 Eutra

SCPI Command :

```
CATalog:SIGNaling:EPS:UECapability:EUTRa:BANDs
```

class EutraCls

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

get_bands() → List[int]

```
# SCPI: CATalog:SIGNaling:EPS:UECapability:EUTRa:BANDs
value: List[int] = driver.catalog.signaling.eps.ueCapability.eutra.get_bands()
```

Queries the list of requested frequency bands configured for the container type 'UE-EUTRA-Capability', for EPS tracking areas.

return
fbi: Comma-separated list of LTE frequency band indicators NAV indicates that there are no requested bands.

6.2.4.1.1.2 Mrdc

SCPI Command :

```
CATalog:SIGNaling:EPS:UECapability:MRDC:BANDs
```

class MrdcCls

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

class BandsStruct

Structure for reading output parameters. Fields:

- Rat: List[enums.CellType]: Type of the band: LTE band or NR band.
- Fbi: List[int]: Frequency band indicator

get_bands() → BandsStruct

```
# SCPI: CATalog:SIGNaling:EPS:UECapability:MRDC:BANDs
value: BandsStruct = driver.catalog.signaling.eps.ueCapability.mrdc.get_bands()
```

Queries the list of requested frequency bands configured for the container type 'UE-MRDC-Capability', for EPS tracking areas. The bands are returned as pairs of values: {<Rat>, <Fbi>}1, {<Rat>, <Fbi>}2, ... A returned pair of NAV indicates that there are no requested bands.

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

6.2.4.1.1.3 Nradio

SCPI Command :

```
CATalog:SIGNaling:EPS:UECapability:NRADio:BANDs
```

class NradioCls

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

get_bands() → List[int]

```
# SCPI: CATalog:SIGNaling:EPS:UECapability:NRADio:BANDs
value: List[int] = driver.catalog.signaling.eps.ueCapability.nradio.get_bands()
```

Queries the list of requested frequency bands configured for the container type 'UE-NR-Capability', for EPS tracking areas.

return
fbi: Comma-separated list of NR frequency band indicators NAV indicates that there are no requested bands.

6.2.4.2 Fgs

class FgsCls

Fgs commands group definition. 3 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.4.2.1 UeCapability

class UeCapabilityCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.catalog.signaling.fgs.ueCapability.clone()
```

Subgroups

6.2.4.2.1.1 Eutra

SCPI Command :

```
CATalog:SIGNaling:FGS:UECapability:EUTRa:BANDs
```

class EutraCls

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

get_bands() → List[int]

```
# SCPI: CATalog:SIGNaling:FGS:UECapability:EUTRa:BANDs
value: List[int] = driver.catalog.signaling.fgs.ueCapability.eutra.get_bands()
```

Queries the list of requested frequency bands configured for the container type 'UE-EUTRA-Capability', for 5GS tracking areas.

return

fbi: Comma-separated list of LTE frequency band indicators NAV indicates that there are no requested bands.

6.2.4.2.1.2 Mrdc

SCPI Command :

```
CATalog:SIGNaling:FGS:UECapability:MRDC:BANDs
```

class MrdcCls

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

class BandsStruct

Structure for reading output parameters. Fields:

- Rat: List[enums.CellType]: Type of the band: LTE band or NR band.
- Fbi: List[int]: Frequency band indicator

get_bands() → BandsStruct

```
# SCPI: CATalog:SIGNaling:FGS:UECapability:MRDC:BANDs
value: BandsStruct = driver.catalog.signaling.fgs.ueCapability.mrdc.get_bands()
```

Queries the list of requested frequency bands configured for the container type 'UE-MRDC-Capability', for 5GS tracking areas. The bands are returned as pairs of values: {<Rat>, <Fbi>}1, {<Rat>, <Fbi>}2, ... A returned pair of NAV indicates that there are no requested bands.

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

6.2.4.2.1.3 Nradio

SCPI Command :

```
CATalog:SIGNaling:FGS:UECapability:NRADio:BANDs
```

class NradioCls

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

get_bands() → List[int]

```
# SCPI: CATalog:SIGNaling:FGS:UECapability:NRADio:BANDs
value: List[int] = driver.catalog.signaling.fgs.ueCapability.nradio.get_bands()
```

Queries the list of requested frequency bands configured for the container type 'UE-NR-Capability', for 5GS tracking areas.

return
fbi: Comma-separated list of NR frequency band indicators NAV indicates that there are no requested bands.

6.2.4.3 Lte

SCPI Commands :

```
CATalog:SIGNaling:LTE:CELL
CATalog:SIGNaling:LTE:CGRoup
```

class LteCls

Lte commands group definition. 6 total commands, 3 Subgroups, 2 group commands

get_cell() → List[str]

```
# SCPI: CATalog:SIGNaling:LTE:CELL
value: List[str] = driver.catalog.signaling.lte.get_cell()
```

Queries a list of all LTE or NR cells.

return
cell_name: Comma-separated list of cell names, one string per cell.

get_cgroup() → List[str]

```
# SCPI: CATalog:SIGNaling:LTE:CGRoup
value: List[str] = driver.catalog.signaling.lte.get_cgroup()
```

Queries a list of all LTE or NR cell groups.

return
cell_group_name: Comma-separated list of cell group names, one string per cell group.

Cloning the Group

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

Subgroups

6.2.4.3.1 Ca

SCPI Command :

```
CATalog:SIGNaling:LTE:CA
```

class CaCls

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

class GetStruct

Response structure. Fields:

- Pcell_Name: str: Name of the PCell or PSCell of the cell group.
- Scell_Name: List[str]: Name of an SCell of the cell group.

get(*cell_group_name: str*) → GetStruct

```
# SCPI: CATalog:SIGNaling:LTE:CA
value: GetStruct = driver.catalog.signaling.lte.ca.get(cell_group_name = 'abc')
```

Queries a list of all cells contained in a specific LTE or NR cell group. The first returned cell is a primary cell. The other cells are secondary cells: <PCellName>, <SCellName>1, ..., <SCellName>n

param cell_group_name
No help available

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

6.2.4.3.2 Ncell

SCPI Command :

```
CATalog:SIGNaling:LTE:NCELl
```

class NcellCls

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

class GetStruct

Response structure. Fields:

- Ncell_Name: List[str]: Neighbor cell name
- Ncell_Type: List[enums.NcellType]: Neighbor cell type IAFRequency: intra-frequency neighbor cell IFRequency: inter-frequency neighbor cell IRAT: inter-RAT neighbor cell

get(*cell_name: str*) → GetStruct

```
# SCPI: CATalog:SIGNaling:LTE:NCELl
value: GetStruct = driver.catalog.signaling.lte.ncell.get(cell_name = 'abc')
```

Queries the SIB neighbor cell list of an LTE or NR cell. For each neighbor cell, two values are returned: {<NCellName>, <NCellType>}cell 1, {<NCellName>, <NCellType>}cell 2, ...

param cell_name
The name of the cell for which neighbors are queried.

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

6.2.4.3.3 Ue

class UeCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.catalog.signaling.lte.ue.clone()
```

Subgroups

6.2.4.3.3.1 Bearer

SCPI Command :

```
CATalog:SIGNaling:LTE:UE:BEARer
```

class BearerCls

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

class GetStruct

Response structure. Fields:

- Linked_Bearer_Id: List[int]: ID of the linked default bearer
- Bearer_Id: List[int]: ID of the dedicated bearer

get(*ue_id: str = None*) → GetStruct

```
# SCPI: CATalog:SIGNaling:LTE:UE:BEARer
value: GetStruct = driver.catalog.signaling.lte.ue.bearer.get(ue_id = 'abc')
```

Queries a list of all established dedicated bearers. For each dedicated bearer, two IDs are returned: {<LinkedBearerId>, <BearerId>}bearer 1, {<LinkedBearerId>, <BearerId>}bearer 2, ...

param ue_id
No help available

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

6.2.4.3.3.2 Dbearer

SCPI Command :

```
CATalog:SIGNaling:LTE:UE:DBEarer
```

class DbearerCls

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

class GetStruct

Response structure. Fields:

- Linked_Bearer_Id: List[int]: ID of the default bearer
- Apn: List[str]: APN of the default bearer

get(*ue_id: str = None*) → GetStruct

```
# SCPI: CATalog:SIGNaling:LTE:UE:DBEarer
value: GetStruct = driver.catalog.signaling.lte.ue.dbearer.get(ue_id = 'abc')
```

Queries a list of all established default bearers. For each default bearer, two values are returned: {<BearerId>, <APN>}bearer 1, {<BearerId>, <APN>}bearer 2, ...

param ue_id
No help available

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

6.2.4.4 Nradio

SCPI Command :

```
CATalog:SIGNaling:NRADio:CGRoup
```

class NradioCls

Nradio commands group definition. 5 total commands, 3 Subgroups, 1 group commands

get_cgroup() → List[str]

```
# SCPI: CATalog:SIGNaling:NRADio:CGRoup
value: List[str] = driver.catalog.signaling.nradio.get_cgroup()
```

Queries a list of all LTE or NR cell groups.

return
cell_group_name: Comma-separated list of cell group names, one string per cell group.

Cloning the Group

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

Subgroups

6.2.4.4.1 Ca

SCPI Command :

```
CATalog:SIGNaling:NRADio:CA
```

class CaCls

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

class GetStruct

Response structure. Fields:

- Pcell_Name: str: Name of the PCell or PSCell of the cell group.
- Scell_Name: List[str]: Name of an SCell of the cell group.

get(*cell_group_name: str*) → GetStruct

```
# SCPI: CATalog:SIGNaling:NRADio:CA
value: GetStruct = driver.catalog.signaling.nradio.ca.get(cell_group_name = 'abc
↪')
```

Queries a list of all cells contained in a specific LTE or NR cell group. The first returned cell is a primary cell. The other cells are secondary cells: <PCellName>, <SCellName>1, ..., <SCellName>n

param cell_group_name
No help available

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

6.2.4.4.2 Cell

SCPI Command :

```
CATalog:SIGNaling:NRADio:CELL
```

class CellCls

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

get_value() → List[str]

```
# SCPI: CATalog:SIGNaling:NRADio:CELL
value: List[str] = driver.catalog.signaling.nradio.cell.get_value()
```

Queries a list of all LTE or NR cells.

return
cell_name: Comma-separated list of cell names, one string per cell.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.catalog.signaling.nradio.cell.clone()
```

Subgroups

6.2.4.4.2.1 Bwp

SCPI Command :

```
CATalog:SIGNaling:NRADio:CELL:BWP
```

class BwpCls

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

get(*cell_name: str*) → List[int]

```
# SCPI: CATalog:SIGNaling:NRADio:CELL:BWP
value: List[int] = driver.catalog.signaling.nradio.cell.bwp.get(cell_name = 'abc
↪')
```

Queries a list of all bandwidth parts.

param cell_name
No help available

return
idn: Comma-separated list of BWP IDs, one ID per BWP. The initial BWP has the ID 0. Additional BWPs have the IDs 1 to n.

6.2.4.4.3 Ncell

SCPI Command :

```
CATalog:SIGNaling:NRADio:NCELl
```

class NcellCls

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

class GetStruct

Response structure. Fields:

- Ncell_Name: List[str]: Neighbor cell name
- Ncell_Type: List[enums.NcellType]: Neighbor cell type IAFRequency: intra-frequency neighbor cell IFRequency: inter-frequency neighbor cell IRAT: inter-RAT neighbor cell

get(*cell_name: str*) → GetStruct

```
# SCPI: CATalog:SIGNaling:NRADio:NCELl
value: GetStruct = driver.catalog.signaling.nradio.ncell.get(cell_name = 'abc')
```

Queries the SIB neighbor cell list of an LTE or NR cell. For each neighbor cell, two values are returned: {<NCellName>, <NCellType>}cell 1, {<NCellName>, <NCellType>}cell 2, ...

param cell_name
The name of the cell for which neighbors are queried.

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

6.2.4.5 Topology

SCPI Command :

```
CATalog:SIGNaling:TOPology:PLMN
```

class TopologyCls

Topology commands group definition. 6 total commands, 2 Subgroups, 1 group commands

get_plmn() → List[str]

```
# SCPI: CATalog:SIGNaling:TOPology:PLMN
value: List[str] = driver.catalog.signaling.topology.get_plmn()
```

Queries a list of all created PLMNs.

return
name_plmn: Comma-separated list of PLMN names, one string per PLMN.

Cloning the Group

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

Subgroups

6.2.4.5.1 Eps

SCPI Commands :

```
CATalog:SIGNaling:TOPology:EPS:UE
CATalog:SIGNaling:TOPology:EPS
```

class EpsCls

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

get(*name_plmn: str = None*) → List[str]

```
# SCPI: CATalog:SIGNaling:TOPology:EPS
value: List[str] = driver.catalog.signaling.topology.eps.get(name_plmn = 'abc')
```

Queries a list of all EPS tracking areas. You can restrict the query to a selected PLMN.

param name_plmn
No help available

return
name_ta_eps: Comma-separated list of tracking area names, one string per tracking area.

get_ue() → List[str]

```
# SCPI: CATalog:SIGNaling:TOPology:EPS:UE
value: List[str] = driver.catalog.signaling.topology.eps.get_ue()
```

No command help available

return
ui_id: No help available

6.2.4.5.2 Fgs

SCPI Command :

```
CATalog:SIGNaling:TOPology:FGS
```

class FgsCls

Fgs commands group definition. 3 total commands, 1 Subgroups, 1 group commands

get(*name_plmn: str = None*) → List[str]

```
# SCPI: CATalog:SIGNaling:TOPology:FGS
value: List[str] = driver.catalog.signaling.topology.fgs.get(name_plmn = 'abc')
```

Queries a list of all 5GS tracking areas. You can restrict the query to a selected PLMN.

param name_plmn
No help available

return
name_ta_5_g: Comma-separated list of tracking area names, one string per tracking area.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.catalog.signaling.topology.fgs.clone()
```

Subgroups

6.2.4.5.2.1 Ue

class UeCls

Ue commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.catalog.signaling.topology.fgs.ue.clone()
```

Subgroups

6.2.4.5.2.2 Pdu

SCPI Command :

```
CATalog:SIGNaling:TOPology:FGS:UE:PDU
```

class PduCls

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

get(*ue_id: str = None*) → List[int]

```
# SCPI: CATalog:SIGNaling:TOPology:FGS:UE:PDU
value: List[int] = driver.catalog.signaling.topology.fgs.ue.pdu.get(ue_id = 'abc
↪')
```

Queries a list of all established PDU sessions.

param ue_id
For future use. Enter any value.

return
pdu_session_id: Comma-separated list of PDU session IDs.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.catalog.signaling.topology.fgs.ue.pdu.clone()
```

Subgroups

6.2.4.5.2.3 QosFlow

SCPI Command :

```
CATalog:SIGNaling:TOPology:FGS:UE:PDU:QOSFlow
```

class QosFlowCls

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

get(*ue_id: str*, *pdu_session_id: int*) → List[int]

```
# SCPI: CATalog:SIGNaling:TOPology:FGS:UE:PDU:QOSFlow
value: List[int] = driver.catalog.signaling.topology.fgs.ue.pdu.qosFlow.get(ue_
↪id = 'abc', pdu_session_id = 1)
```

Queries a list of all QoS flows of a PDU session.

param ue_id
For future use. Enter any value.

param pdu_session_id
ID of the PDU session for which the QoS flows are queried.

return
qos_flow_id: Comma-separated list of QoS flow IDs.

6.2.4.6 Trigger

SCPI Command :

```
CATalog:SIGNaling:TRIGger:SOURce
```

class TriggerCls

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

get_source() → List[str]

```
# SCPI: CATalog:SIGNaling:TRIGger:SOURce
value: List[str] = driver.catalog.signaling.trigger.get_source()
```

Queries a list of all inactive trigger types that can be activated using method RsCMX_Signaling.Configure.Signaling. Trigger.scope.

return
trigger: Comma-separated list of strings, one string per trigger type.

6.2.5 Wlan

class WlanCls

Wlan commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.5.1 Measurement<MeasInstance>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.catalog.wlan.measurement.repcap_measInstance_get()
driver.catalog.wlan.measurement.repcap_measInstance_set(repcap.MeasInstance.Nr1)
```

class MeasurementCls

Measurement commands group definition. 2 total commands, 1 Subgroups, 0 group commands Repeated Capability: MeasInstance, default value after init: MeasInstance.Nr1

Cloning the Group

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

Subgroups

6.2.5.1.1 Network

class NetworkCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.catalog.wlan.measurement.network.clone()
```

Subgroups

6.2.5.1.1.1 Cell

class CellCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.catalog.wlan.measurement.network.cell.clone()
```

Subgroups

6.2.5.1.1.2 Uplinks

SCPI Command :

```
CATalog:WLAN:MEASurement<Instance>:NETWork:CELL:UPLinks
```

class UplinksCls

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

get(*cell_name: str*, *measInstance=MeasInstance.Default*) → List[int]

```
# SCPI: CATalog:WLAN:MEASurement<Instance>:NETWork:CELL:UPLinks
value: List[int] = driver.catalog.wlan.measurement.network.cell.uplinks.
↪get(cell_name = 'abc', measInstance = repcap.MeasInstance.Default)
```

No command help available

param cell_name
No help available

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

return
available_ul: No help available

6.2.5.1.1.3 Cells

SCPI Command :

```
CATalog:WLAN:MEASurement<Instance>:NETWork:CELLs
```

class CellsCls

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

get(*measInstance=MeasInstance.Default*) → List[str]

```
# SCPI: CATalog:WLAN:MEASurement<Instance>:NETWork:CELLs
value: List[str] = driver.catalog.wlan.measurement.network.cells.
↪get(measInstance = repcap.MeasInstance.Default)
```

No command help available

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

return
cell_name: No help available

6.3 Configure

class ConfigureCls

Configure commands group definition. 903 total commands, 5 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.1 Lte

class LteCls

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

Cloning the Group

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

Subgroups

6.3.1.1 Measurement<MeasInstance>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.configure.lte.measurement.repcap_measInstance_get()
driver.configure.lte.measurement.repcap_measInstance_set(repcap.MeasInstance.Nr1)
```

class MeasurementCls

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

Cloning the Group

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

Subgroups

6.3.1.1.1 Network

class NetworkCls

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

Cloning the Group

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

Subgroups

6.3.1.1.1.1 Cell

SCPI Command :

```
[CONFigure]:LTE:MEASurement<Instance>:NETWork:CELL
```

class CellCls

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

class CellStruct

Response structure. Fields:

- Cell_Name: str: No parameter help available
- Used_Ul: List[int]: No parameter help available

get(*measInstance=MeasInstance.Default*) → CellStruct

```
# SCPI: [CONFigure]:LTE:MEASurement<Instance>:NETWork:CELL
value: CellStruct = driver.configure.lte.measurement.network.cell.
↪get(measInstance = repcap.MeasInstance.Default)
```

No command help available

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

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

set(*cell_name: str*, *used_ul: List[int] = None*, *measInstance=MeasInstance.Default*) → None

```
# SCPI: [CONFigure]:LTE:MEASurement<Instance>:NETWork:CELL
driver.configure.lte.measurement.network.cell.set(cell_name = 'abc', used_ul =␣
↪[1, 2, 3], measInstance = repcap.MeasInstance.Default)
```

No command help available

param cell_name
No help available

param used_ul
No help available

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

6.3.2 NrMmw

class NrMmwCls

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

Cloning the Group

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

Subgroups

6.3.2.1 Measurement<MeasInstance>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.configure.nrMmw.measurement.repcap_measInstance_get()
driver.configure.nrMmw.measurement.repcap_measInstance_set(repcap.MeasInstance.Nr1)
```

class MeasurementCls

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

Cloning the Group

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

Subgroups

6.3.2.1.1 Network

class NetworkCls

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

Cloning the Group

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

Subgroups

6.3.2.1.1.1 Cell

SCPI Command :

```
[CONFigure]:NRMMw:MEASurement<Instance>:NETWork:CELL
```

class CellCls

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

class CellStruct

Response structure. Fields:

- Cell_Name: str: No parameter help available
- Used_Ul: List[int]: No parameter help available

get(*measInstance=MeasInstance.Default*) → CellStruct

```
# SCPI: [CONFigure]:NRMMw:MEASurement<Instance>:NETWork:CELL
value: CellStruct = driver.configure.nrMmw.measurement.network.cell.
↪get(measInstance = repcap.MeasInstance.Default)
```

No command help available

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

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

set(*cell_name: str*, *used_ul: List[int] = None*, *measInstance=MeasInstance.Default*) → None

```
# SCPI: [CONFigure]:NRMMw:MEASurement<Instance>:NETWork:CELL
driver.configure.nrMmw.measurement.network.cell.set(cell_name = 'abc', used_ul␣
↪= [1, 2, 3], measInstance = repcap.MeasInstance.Default)
```

No command help available

param cell_name
No help available

param used_ul
No help available

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

6.3.3 NrSub

class NrSubCls

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

Cloning the Group

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

Subgroups

6.3.3.1 Measurement<MeasInstance>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.configure.nrSub.measurement.repcap_measInstance_get()
driver.configure.nrSub.measurement.repcap_measInstance_set(repcap.MeasInstance.Nr1)
```

class MeasurementCls

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

Cloning the Group

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

Subgroups

6.3.3.1.1 Network

class NetworkCls

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

Cloning the Group

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

Subgroups

6.3.3.1.1.1 Cell

SCPI Command :

```
[CONFigure]:NRSub:MEASurement<Instance>:NETWork:CELL
```

class CellCls

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

class CellStruct

Response structure. Fields:

- Cell_Name: str: No parameter help available
- Used_Ul: List[int]: No parameter help available

get(*measInstance=MeasInstance.Default*) → CellStruct

```
# SCPI: [CONFigure]:NRSub:MEASurement<Instance>:NETWork:CELL
value: CellStruct = driver.configure.nrSub.measurement.network.cell.
↪get(measInstance = repcap.MeasInstance.Default)
```

No command help available

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

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

set(*cell_name: str*, *used_ul: List[int] = None*, *measInstance=MeasInstance.Default*) → None

```
# SCPI: [CONFigure]:NRSub:MEASurement<Instance>:NETWork:CELL
driver.configure.nrSub.measurement.network.cell.set(cell_name = 'abc', used_ul␣
↪= [1, 2, 3], measInstance = repcap.MeasInstance.Default)
```

No command help available

param cell_name
No help available

param used_ul
No help available

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

6.3.4 Signaling

SCPI Commands :

```
[CONFigure]:SIGNaling:MCGRoup
[CONFigure]:SIGNaling:SCGRoup
[CONFigure]:SIGNaling:APMod
```

class SignalingCls

Signaling commands group definition. 899 total commands, 16 Subgroups, 3 group commands

get_ap_mod() → bool

```
# SCPI: [CONFigure]:SIGNaling:APMod
value: bool = driver.configure.signaling.get_ap_mod()
```

Selects whether changes are applied automatically. Applying several commands simultaneously can be necessary to avoid call drops caused by intermediate combinations of settings.

return
enable: - ON: If you modify settings via a command, the changes are applied immediately (known default behavior) . - OFF: Changes are not applied automatically. Use the command PROCedure:SIGNaling:APMod to apply pending changes.This behavior corresponds to a dialog box with an apply button, where you apply several changes simultaneously.

get_mc_group() → str

```
# SCPI: [CONFigure]:SIGNaling:MCGRoup
value: str = driver.configure.signaling.get_mc_group()
```

No command help available

return
cell_group_name: No help available

get_sc_group() → str

```
# SCPI: [CONFigure]:SIGNaling:SCGRoup
value: str = driver.configure.signaling.get_sc_group()
```

No command help available

return
cell_group_name: No help available

set_ap_mod(*enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:APMod
driver.configure.signaling.set_ap_mod(enable = False)
```

Selects whether changes are applied automatically. Applying several commands simultaneously can be necessary to avoid call drops caused by intermediate combinations of settings.

param enable

- ON: If you modify settings via a command, the changes are applied immediately (known default behavior) .
- OFF: Changes are not applied automatically. Use the command PROCedure:SIGNaling:APMod to apply pending changes.This behavior corresponds to a dialog box with an apply button, where you apply several changes simultaneously.

Cloning the Group

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

Subgroups

6.3.4.1 Awgn

class AwgnCls

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

Cloning the Group

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

Subgroups

6.3.4.1.1 Advanced

class AdvancedCls

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

Cloning the Group

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

Subgroups

6.3.4.1.1.1 Bandwidth

class BandwidthCls

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

Cloning the Group

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

Subgroups

6.3.4.1.1.2 Ratio

SCPI Command :

```
[CONFigure]:SIGNaling:AWGN:ADVanced:BWIDth:RATio
```

class RatioCls

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

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:AWGN:ADVanced:BWIDth:RATio
value: float = driver.configure.signaling.awgn.advanced.bandwidth.ratio.
↪get(cell_name = 'abc')
```

Configures the minimum ratio between the noise bandwidth and the carrier bandwidth for AWGN insertion.

param cell_name
No help available

return
ratio: No help available

set(*cell_name: str*, *ratio: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:AWGN:ADVanced:BWIDth:RATio
driver.configure.signaling.awgn.advanced.bandwidth.ratio.set(cell_name = 'abc',␣
↪ratio = 1.0)
```

Configures the minimum ratio between the noise bandwidth and the carrier bandwidth for AWGN insertion.

param cell_name
No help available

param ratio
No help available

6.3.4.1.2 Enable

SCPI Command :

```
[CONFigure]:SIGNaling:AWGN:ENABle
```

class EnableCls

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

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:AWGN:ENABle
value: bool = driver.configure.signaling.awgn.enable.get(cell_name = 'abc')
```

Enables AWGN insertion.

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:AWGN:ENABle
driver.configure.signaling.awgn.enable.set(cell_name = 'abc', enable = False)
```

Enables AWGN insertion.

param cell_name
No help available

param enable
No help available

6.3.4.1.3 SnRatio

SCPI Command :

```
[CONFigure]:SIGNaling:AWGN:SNRatio
```

class SnRatioCls

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

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:AWGN:SNRatio
value: float = driver.configure.signaling.awgn.snRatio.get(cell_name = 'abc')
```

Specifies the signal to noise ratio for AWGN insertion.

param cell_name
No help available

return
ratio: No help available

set(*cell_name: str*, *ratio: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:AWGN:SNRatio
driver.configure.signaling.awgn.snRatio.set(cell_name = 'abc', ratio = 1.0)
```

Specifies the signal to noise ratio for AWGN insertion.

param cell_name
No help available

param ratio
No help available

6.3.4.2 Cmas

class CmasCls

Cmas commands group definition. 7 total commands, 7 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.4.2.1 Cgroup

SCPI Command :

```
[CONFigure]:SIGNaling:CMAS:CGRoup
```

class CgroupCls

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

get(*network_scope: str*) → CodingGroup

```
# SCPI: [CONFigure]:SIGNaling:CMAS:CGRoup
value: enums.CodingGroup = driver.configure.signaling.cmas.cgroup.get(network_
↪scope = 'abc')
```

Selects the coding group for CMAS messages.

param network_scope
No help available

return
coding_group: G7: GSM 7-bit coding without language string G7L: GSM 7-bit coding with language string U2L: UCS-2 coding with language string

set(*network_scope: str*, *coding_group: CodingGroup*) → None

```
# SCPI: [CONFigure]:SIGNaling:CMAS:CGRoup
driver.configure.signaling.cmas.cgroup.set(network_scope = 'abc', coding_group␣
↪= enums.CodingGroup.G7)
```

Selects the coding group for CMAS messages.

param network_scope
No help available

param coding_group
G7: GSM 7-bit coding without language string G7L: GSM 7-bit coding with language string U2L: UCS-2 coding with language string

6.3.4.2.2 Data

SCPI Command :

```
[CONFigure]:SIGNaling:CMAS:DATA
```

class DataCls

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

get(*network_scope: str*) → str

```
# SCPI: [CONFigure]:SIGNaling:CMAS:DATA
value: str = driver.configure.signaling.cmas.data.get(network_scope = 'abc')
```

Defines the broadcasted CMAS message text.

param network_scope
No help available

return
data: No help available

set(*network_scope: str*, *data: str*) → None

```
# SCPI: [CONFigure]:SIGNaling:CMAS:DATA
driver.configure.signaling.cmas.data.set(network_scope = 'abc', data = 'abc')
```

Defines the broadcasted CMAS message text.

param network_scope
No help available

param data
No help available

6.3.4.2.3 Id

SCPI Command :

```
[CONFigure]:SIGNaling:CMAS:ID
```

class IdCls

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

get(*network_scope: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:CMAS:ID
value: int = driver.configure.signaling.cmas.id.get(network_scope = 'abc')
```

Defines the message identifier for CMAS messages.

param network_scope
No help available

return
identifier: No help available

set(*network_scope: str*, *identifier: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:CMAS:ID
driver.configure.signaling.cmas.id.set(network_scope = 'abc', identifier = 1)
```

Defines the message identifier for CMAS messages.

param network_scope
No help available

param identifier
No help available

6.3.4.2.4 Language

SCPI Command :

```
[CONFigure]:SIGNaling:CMAS:LANGuage
```

class LanguageCls

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

class GetStruct

Response structure. Fields:

- Language: str: Two-character language string
- Cgroup_Language: enums.GroupLanguage: Selects the coding group (GSM 7-bit or UCS-2) . Omitting the value sets G7L.

get(*network_scope: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:CMAS:LANGuage
value: GetStruct = driver.configure.signaling.cmas.language.get(network_scope =
↪'abc')
```

Selects the language string and sets the coding group for CMAS messages.

param network_scope
No help available

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

set(*network_scope: str*, *language: str*, *cgroup_language: GroupLanguage = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:CMAS:LANGuage
driver.configure.signaling.cmas.language.set(network_scope = 'abc', language =
↪'abc', cgroup_language = enums.GroupLanguage.G7L)
```

Selects the language string and sets the coding group for CMAS messages.

param network_scope
No help available

param language
Two-character language string

param cgroup_language
Selects the coding group (GSM 7-bit or UCS-2) . Omitting the value sets G7L.

6.3.4.2.5 Serial

SCPI Command :

```
[CONFigure]:SIGNaling:CMAS:SERial
```

class SerialCls

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

class GetStruct

Response structure. Fields:

- Display_Mode: enums.DisplayMode: IMMediate: The UE displays a message immediately. NORMal: The UE displays a message upon user action.
- Message_Code: int: The last 8 bits of the message code of the serial number.
- Update_Time: int: The version of the message transmitted as an update number.

get(*network_scope: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:CMAS:SERial
value: GetStruct = driver.configure.signaling.cmas.serial.get(network_scope =
↪'abc')
```

Defines settings influencing the serial number for CMAS messages.

param network_scope
No help available

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

set(*network_scope: str*, *display_mode: DisplayMode*, *message_code: int = None*, *update_time: int = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:CMAS:SERial
driver.configure.signaling.cmas.serial.set(network_scope = 'abc', display_mode␣
↪= enums.DisplayMode.IMMediate, message_code = 1, update_time = 1)
```

Defines settings influencing the serial number for CMAS messages.

param network_scope
No help available

param display_mode
IMMediate: The UE displays a message immediately. NORMal: The UE displays a message upon user action.

param message_code
The last 8 bits of the message code of the serial number.

param update_time
The version of the message transmitted as an update number.

6.3.4.2.6 Tranmission

SCPI Command :

```
[CONFigure]:SIGNaling:CMAS:TRANmission
```

class TranmissionCls

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

get(*network_scope: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:CMAS:TRANmission
value: bool = driver.configure.signaling.cmas.tranmission.get(network_scope =
↪'abc')
```

Enables or disables the transmission of CMAS messages via the SIB.

param network_scope
No help available

return
enable: No help available

set(*network_scope: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:CMAS:TRANmission
driver.configure.signaling.cmas.tranmission.set(network_scope = 'abc', enable =␣
↪False)
```

Enables or disables the transmission of CMAS messages via the SIB.

param network_scope
No help available

param enable
No help available

6.3.4.2.7 WoLanguage

SCPI Command :

```
[CONFigure]:SIGNaling:CMAS:WOLanguage
```

class WoLanguageCls

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

get(*network_scope: str*) → LanguageB

```
# SCPI: [CONFigure]:SIGNaling:CMAS:WOLanguage
value: enums.LanguageB = driver.configure.signaling.cmas.woLanguage.get(network_
↪scope = 'abc')
```

Selects the language and sets the coding group 'GSM 7-bit coding without language string', for CMAS messages.

param network_scope
No help available

return
language: No help available

set(*network_scope: str*, *language: LanguageB*) → None

```
# SCPI: [CONFigure]:SIGNaling:CMAS:WOLanguage
driver.configure.signaling.cmas.woLanguage.set(network_scope = 'abc', language
↪= enums.LanguageB.DANish)
```

Selects the language and sets the coding group 'GSM 7-bit coding without language string', for CMAS messages.

param network_scope
No help available

param language
No help available

6.3.4.3 Eps

SCPI Command :

```
[CONFigure]:SIGNaling:EPS:TMODe
```

class EpsCls

Eps commands group definition. 22 total commands, 5 Subgroups, 1 group commands

get_tmode() → bool

```
# SCPI: [CONFigure]:SIGNaling:EPS:TMODe
value: bool = driver.configure.signaling.eps.get_tmode()
```

Selects whether an 'ACTIVATE TEST MODE' message is sent to the UE during registration in an EPS tracking area, or not. Configure this setting before registration.

return
enable: No help available

set_tmode(*enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:EPS:TMODe
driver.configure.signaling.eps.set_tmode(enable = False)
```

Selects whether an 'ACTIVATE TEST MODE' message is sent to the UE during registration in an EPS tracking area, or not. Configure this setting before registration.

param enable
No help available

Cloning the Group

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

Subgroups

6.3.4.3.1 AsPy

class AsPyCls

AsPy commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.4.3.1.1 Security

SCPI Commands :

```
[CONFigure]:SIGNaling:EPS:AS:SECurity:INTegrity
[CONFigure]:SIGNaling:EPS:AS:SECurity:CIPHering
```

class SecurityCls

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

get_ciphering() → SecurityAlgorithm

```
# SCPI: [CONFigure]:SIGNaling:EPS:AS:SECurity:CIPHering
value: enums.SecurityAlgorithm = driver.configure.signaling.eps.asPy.security.
↪get_ciphering()
```

Selects an algorithm for AS ciphering in EPS tracking areas.

return
algorithm: No help available

get_integrity() → SecurityAlgorithm

```
# SCPI: [CONFigure]:SIGNaling:EPS:AS:SECurity:INTegrity
value: enums.SecurityAlgorithm = driver.configure.signaling.eps.asPy.security.
↪get_integrity()
```

Selects an algorithm for AS integrity protection in EPS tracking areas.

return
algorithm: No help available

set_ciphering(*algorithm: SecurityAlgorithm*) → None

```
# SCPI: [CONFigure]:SIGNaling:EPS:AS:SECurity:CIPHering
driver.configure.signaling.eps.asPy.security.set_ciphering(algorithm = enums.
↪SecurityAlgorithm.AES)
```

Selects an algorithm for AS ciphering in EPS tracking areas.

param algorithm
No help available

set_integrity(*algorithm: SecurityAlgorithm*) → None

```
# SCPI: [CONFigure]:SIGNaling:EPS:AS:SECurity:INTegrity
driver.configure.signaling.eps.asPy.security.set_integrity(algorithm = enums.
↪SecurityAlgorithm.AES)
```

Selects an algorithm for AS integrity protection in EPS tracking areas.

param algorithm
No help available

6.3.4.3.2 Dbearer

SCPI Commands :

```
[CONFigure]:SIGNaling:EPS:DBEarer:APN
[CONFigure]:SIGNaling:EPS:DBEarer:RLCMode
```

class DbearerCls

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

get_apn() → str

```
# SCPI: [CONFigure]:SIGNaling:EPS:DBEarer:APN
value: str = driver.configure.signaling.eps.dbearer.get_apn()
```

Configures the default APN for default bearers in EPS tracking areas.

return
apn: No help available

get_rlc_mode() → RlcMode

```
# SCPI: [CONFigure]:SIGNaling:EPS:DBEarer:RLCMode
value: enums.RlcMode = driver.configure.signaling.eps.dbearer.get_rlc_mode()
```

Configures the RLC mode for default bearers in EPS tracking areas.

return
rlc_mode: RLC mode ACK: acknowledged UACK: unacknowledged

set_apn(*apn: str*) → None

```
# SCPI: [CONFigure]:SIGNaling:EPS:DBEarer:APN
driver.configure.signaling.eps.dbearer.set_apn(apn = 'abc')
```

Configures the default APN for default bearers in EPS tracking areas.

param apn
No help available

`set_rlc_mode`(*rlc_mode: RlcMode*) → None

```
# SCPI: [CONFigure]:SIGNaling:EPS:DBEarer:RLCMode
driver.configure.signaling.eps.dbearer.set_rlc_mode(rlc_mode = enums.RlcMode.
↪ACK)
```

Configures the RLC mode for default bearers in EPS tracking areas.

param rlc_mode
RLC mode ACK: acknowledged UACK: unacknowledged

6.3.4.3.3 Nas

class NasCls

Nas commands group definition. 9 total commands, 3 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.4.3.3.1 Auth

SCPI Commands :

```
[CONFigure]:SIGNaling:EPS:NAS:AUTH:ENABle
[CONFigure]:SIGNaling:EPS:NAS:AUTH:RAND
[CONFigure]:SIGNaling:EPS:NAS:AUTH:IRES
```

class AuthCls

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

`get_enable`() → bool

```
# SCPI: [CONFigure]:SIGNaling:EPS:NAS:AUTH:ENABle
value: bool = driver.configure.signaling.eps.nas.auth.get_enable()
```

Enables authentication for EPS tracking areas.

return
enable: No help available

`get_ires`() → bool

```
# SCPI: [CONFigure]:SIGNaling:EPS:NAS:AUTH:IRES
value: bool = driver.configure.signaling.eps.nas.auth.get_ires()
```

Enables ignoring the RES in EPS tracking areas (successful authentication, even if the UE returns a wrong RES value) .

return
enable: No help available

get_rand() → str

```
# SCPI: [CONFigure]:SIGNaling:EPS:NAS:AUTH:RAND
value: str = driver.configure.signaling.eps.nas.auth.get_rand()
```

Defines the random number (RAND) to be used for authentication in EPS tracking areas.

return
rand: No help available

set_enable(*enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:EPS:NAS:AUTH:ENABle
driver.configure.signaling.eps.nas.auth.set_enable(enable = False)
```

Enables authentication for EPS tracking areas.

param enable
No help available

set_ires(*enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:EPS:NAS:AUTH:IRES
driver.configure.signaling.eps.nas.auth.set_ires(enable = False)
```

Enables ignoring the RES in EPS tracking areas (successful authentication, even if the UE returns a wrong RES value) .

param enable
No help available

set_rand(*rand: str*) → None

```
# SCPI: [CONFigure]:SIGNaling:EPS:NAS:AUTH:RAND
driver.configure.signaling.eps.nas.auth.set_rand(rand = 'abc')
```

Defines the random number (RAND) to be used for authentication in EPS tracking areas.

param rand
No help available

6.3.4.3.3.2 Security

SCPI Commands :

```
[CONFigure]:SIGNaling:EPS:NAS:SECurity:ENABle
[CONFigure]:SIGNaling:EPS:NAS:SECurity:INTegrity
[CONFigure]:SIGNaling:EPS:NAS:SECurity:CIPHering
```

class SecurityCls

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

get_ciphering() → SecurityAlgorithmB

```
# SCPI: [CONFigure]:SIGNaling:EPS:NAS:SECurity:CIPHering
value: enums.SecurityAlgorithmB = driver.configure.signaling.eps.nas.security.
↪get_ciphering()
```

Selects an algorithm for NAS ciphering in EPS tracking areas.

return
algorithm: No help available

get_enable() → bool

```
# SCPI: [CONFigure]:SIGNaling:EPS:NAS:SECurity:ENABle
value: bool = driver.configure.signaling.eps.nas.security.get_enable()
```

Enables security procedures (ciphering, integrity protection) for EPS tracking areas.

return
enable: No help available

get_integrity() → SecurityAlgorithmC

```
# SCPI: [CONFigure]:SIGNaling:EPS:NAS:SECurity:INTegrity
value: enums.SecurityAlgorithmC = driver.configure.signaling.eps.nas.security.
↪get_integrity()
```

Selects an algorithm for NAS integrity protection in EPS tracking areas.

return
algorithm: No help available

set_ciphering(*algorithm: SecurityAlgorithmB*) → None

```
# SCPI: [CONFigure]:SIGNaling:EPS:NAS:SECurity:CIPHering
driver.configure.signaling.eps.nas.security.set_ciphering(algorithm = enums.
↪SecurityAlgorithmB.EEA0)
```

Selects an algorithm for NAS ciphering in EPS tracking areas.

param algorithm
No help available

set_enable(*enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:EPS:NAS:SECurity:ENABle
driver.configure.signaling.eps.nas.security.set_enable(enable = False)
```

Enables security procedures (ciphering, integrity protection) for EPS tracking areas.

param enable
No help available

set_integrity(*algorithm: SecurityAlgorithmC*) → None

```
# SCPI: [CONFigure]:SIGNaling:EPS:NAS:SECurity:INTegrity
driver.configure.signaling.eps.nas.security.set_integrity(algorithm = enums.
↪SecurityAlgorithmC.EIA0)
```

Selects an algorithm for NAS integrity protection in EPS tracking areas.

param algorithm
No help available

6.3.4.3.3.3 Tlv

SCPI Commands :

```
[CONFigure]:SIGNaling:EPS:NAS:TLV:ATTaccept
[CONFigure]:SIGNaling:EPS:NAS:TLV:DBEarer
[CONFigure]:SIGNaling:EPS:NAS:TLV:BEARer
```

class TlvCls

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

set_att_accept(*message: str*) → None

```
# SCPI: [CONFigure]:SIGNaling:EPS:NAS:TLV:ATTaccept
driver.configure.signaling.eps.nas.tlv.set_att_accept(message = 'abc')
```

No command help available

param message
No help available

set_bearer(*message: str*) → None

```
# SCPI: [CONFigure]:SIGNaling:EPS:NAS:TLV:BEARer
driver.configure.signaling.eps.nas.tlv.set_bearer(message = 'abc')
```

No command help available

param message
No help available

set_dbearer(*message: str*) → None

```
# SCPI: [CONFigure]:SIGNaling:EPS:NAS:TLV:DBEarer
driver.configure.signaling.eps.nas.tlv.set_dbearer(message = 'abc')
```

No command help available

param message
No help available

6.3.4.3.4 Nbehavior

SCPI Commands :

```
[CONFigure]:SIGNaling:EPS:NBEHavior:DITimer
[CONFigure]:SIGNaling:EPS:NBEHavior:KRRC
```

class NbehaviorCls

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

get_di_timer() → float

```
# SCPI: [CONFigure]:SIGNaling:EPS:NBEHavior:DITimer
value: float or bool = driver.configure.signaling.eps.nbehavior.get_di_timer()
```

Configures the data inactivity timer for EPS tracking areas. With enabled timer, an RRC connection is released when there has been no activity on the connection (no traffic) for the configured time.

return
timer: (float or boolean) Numeric value: Enables the timer and sets the timer value. ON: Enables the timer, using the configured numeric value. OFF: Disables the timer (no release due to inactivity) .

get_krrc() → bool

```
# SCPI: [CONFigure]:SIGNaling:EPS:NBEHavior:KRRC
value: bool = driver.configure.signaling.eps.nbehavior.get_krrc()
```

Selects whether the RRC connection is kept after a registration in an EPS tracking area.

return
enable: No help available

set_di_timer(*timer: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:EPS:NBEHavior:DITimer
driver.configure.signaling.eps.nbehavior.set_di_timer(timer = 1.0)
```

Configures the data inactivity timer for EPS tracking areas. With enabled timer, an RRC connection is released when there has been no activity on the connection (no traffic) for the configured time.

param timer
(float or boolean) Numeric value: Enables the timer and sets the timer value. ON: Enables the timer, using the configured numeric value. OFF: Disables the timer (no release due to inactivity) .

set_krrc(*enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:EPS:NBEHavior:KRRC
driver.configure.signaling.eps.nbehavior.set_krrc(enable = False)
```

Selects whether the RRC connection is kept after a registration in an EPS tracking area.

param enable
No help available

Cloning the Group

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

Subgroups

6.3.4.3.4.1 RrcReject

SCPI Command :

```
[CONFigure]:SIGNaling:EPS:NBEHavior:RRCReject
```

class RrcRejectCls

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

class RrcRejectStruct

Response structure. Fields:

- Reject_Procedure: enums.EpsRejectProcedure: Selects the procedure to fail. NOR: No reject. ATTR: 'Attach Reject' message PDNR: 'PDN Connectivity Reject' message TAUR: 'Tracking Area Update Reject' message BEAR: 'Bearer Resource Allocation Reject' message
- Reject_Cause: enums.EpsRejectCause: Selects the cause to be signaled to the UE as the reason for the rejection.

get() → RrcRejectStruct

```
# SCPI: [CONFigure]:SIGNaling:EPS:NBEHavior:RRCReject
value: RrcRejectStruct = driver.configure.signaling.eps.nbehavior.rrcReject.
↪get()
```

Configures RRC reject causes for EPS tracking areas.

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

set(*reject_procedure: EpsRejectProcedure*, *reject_cause: EpsRejectCause = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:EPS:NBEHavior:RRCReject
driver.configure.signaling.eps.nbehavior.rrcReject.set(reject_procedure = enums.
↪EpsRejectProcedure.ATTR, reject_cause = enums.EpsRejectCause.C002)
```

Configures RRC reject causes for EPS tracking areas.

param reject_procedure
Selects the procedure to fail. NOR: No reject. ATTR: 'Attach Reject' message PDNR: 'PDN Connectivity Reject' message TAUR: 'Tracking Area Update Reject' message BEAR: 'Bearer Resource Allocation Reject' message

param reject_cause
Selects the cause to be signaled to the UE as the reason for the rejection.

6.3.4.3.5 UeCapability

SCPI Commands :

```
[CONFigure]:SIGNaling:EPS:UECapability:MODE
[CONFigure]:SIGNaling:EPS:UECapability:SEGMentation
```

class UeCapabilityCls

UeCapability commands group definition. 5 total commands, 2 Subgroups, 2 group commands

get_mode() → ModeUeCapability

```
# SCPI: [CONFigure]:SIGNaling:EPS:UECapability:MODE
value: enums.ModeUeCapability = driver.configure.signaling.eps.ueCapability.get_
↪mode()
```

Selects the configuration mode for 'UeCapabilityEnquiry' messages in EPS tracking areas.

return
mode: SKIP: no 'UeCapabilityEnquiry' messages AUTO: automatic message configuration UDEFined: configuration via the other commands in this chapter

get_segmentation() → bool

```
# SCPI: [CONFigure]:SIGNaling:EPS:UECapability:SEGMentation
value: bool = driver.configure.signaling.eps.ueCapability.get_segmentation()
```

Selects whether the UE is allowed to use segmentation for capability information in EPS tracking areas.

return
enable: No help available

set_mode(*mode: ModeUeCapability*) → None

```
# SCPI: [CONFigure]:SIGNaling:EPS:UECapability:MODE
driver.configure.signaling.eps.ueCapability.set_mode(mode = enums.
↪ModeUeCapability.AUTO)
```

Selects the configuration mode for 'UeCapabilityEnquiry' messages in EPS tracking areas.

param mode
SKIP: no 'UeCapabilityEnquiry' messages AUTO: automatic message configuration UDEFined: configuration via the other commands in this chapter

set_segmentation(*enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:EPS:UECapability:SEGMentation
driver.configure.signaling.eps.ueCapability.set_segmentation(enable = False)
```

Selects whether the UE is allowed to use segmentation for capability information in EPS tracking areas.

param enable
No help available

Cloning the Group

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

Subgroups

6.3.4.3.5.1 Eutra

SCPI Commands :

```
[CONFigure]:SIGNaling:EPS:UECapability:EUTRa:RFORmat
[CONFigure]:SIGNaling:EPS:UECapability:EUTRa:SFC
```

class EutraCls

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

get_rformat() → bool

```
# SCPI: [CONFigure]:SIGNaling:EPS:UECapability:EUTRa:RFORmat
value: bool = driver.configure.signaling.eps.ueCapability.eutra.get_rformat()
```

Adds the field 'requestReducedFormat-r13' to the message 'UeCapabilityEnquiry', for EPS tracking areas, container type 'UE-EUTRA-Capability'.

return
enable: No help available

get_sfc() → bool

```
# SCPI: [CONFigure]:SIGNaling:EPS:UECapability:EUTRa:SFC
value: bool = driver.configure.signaling.eps.ueCapability.eutra.get_sfc()
```

Adds the field 'requestSkipFallbackComb-r13' to the message 'UeCapabilityEnquiry', for EPS tracking areas, container type 'UE-EUTRA-Capability'.

return
enable: No help available

set_rformat(*enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:EPS:UECapability:EUTRa:RFORmat
driver.configure.signaling.eps.ueCapability.eutra.set_rformat(enable = False)
```

Adds the field 'requestReducedFormat-r13' to the message 'UeCapabilityEnquiry', for EPS tracking areas, container type 'UE-EUTRA-Capability'.

param enable
No help available

set_sfc(*enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:EPS:UECapability:EUTRa:SFC
driver.configure.signaling.eps.ueCapability.eutra.set_sfc(enable = False)
```

Adds the field 'requestSkipFallbackComb-r13' to the message 'UeCapabilityEnquiry', for EPS tracking areas, container type 'UE-EUTRA-Capability'.

param enable
No help available

6.3.4.3.5.2 Mrdc

SCPI Command :

```
[CONFigure]:SIGNaling:EPS:UECapability:MRDC:ENRonly
```

class MrdcCls

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

get_enr_only() → bool

```
# SCPI: [CONFigure]:SIGNaling:EPS:UECapability:MRDC:ENRonly
value: bool = driver.configure.signaling.eps.ueCapability.mrdc.get_enr_only()
```

Adds the field 'eutra-nr-only-r15' to the message 'UeCapabilityEnquiry', for EPS tracking areas, container type 'UE-MRDC-Capability'.

return
enable: No help available

set_enr_only(*enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:EPS:UECapability:MRDC:ENRonly
driver.configure.signaling.eps.ueCapability.mrdc.set_enr_only(enable = False)
```

Adds the field 'eutra-nr-only-r15' to the message 'UeCapabilityEnquiry', for EPS tracking areas, container type 'UE-MRDC-Capability'.

param enable
No help available

6.3.4.4 Etws

class EtwsCls

Etws commands group definition. 12 total commands, 6 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.4.4.1 Alert

SCPI Command :

```
[CONFigure]:SIGNaling:ETWS:ALERt
```

class AlertCls

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

get(*network_scope: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:ETWS:ALERt
value: bool = driver.configure.signaling.etws.alert.get(network_scope = 'abc')
```

Enables an emergency user alert at the UE, for ETWS primary notifications.

param network_scope
No help available

return
enable: No help available

set(*network_scope: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:ETWS:ALERt
driver.configure.signaling.etws.alert.set(network_scope = 'abc', enable = False)
```

Enables an emergency user alert at the UE, for ETWS primary notifications.

param network_scope
No help available

param enable
No help available

6.3.4.4.2 Id

SCPI Command :

```
[CONFigure]:SIGNaling:ETWS:ID
```

class IdCls

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

get(*network_scope: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:ETWS:ID
value: int = driver.configure.signaling.etws.id.get(network_scope = 'abc')
```

Defines the message identifier for ETWS primary notifications.

param network_scope
No help available

return
identifier: No help available

set(*network_scope: str*, *identifier: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:ETWS:ID
driver.configure.signaling.etws.id.set(network_scope = 'abc', identifier = 1)
```

Defines the message identifier for ETWS primary notifications.

param network_scope
No help available

param identifier
No help available

6.3.4.4.3 Popup

SCPI Command :

```
[CONFigure]:SIGNaling:ETWS:POPup
```

class PopupCls

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

get(*network_scope: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:ETWS:POPup
value: bool = driver.configure.signaling.etws.popup.get(network_scope = 'abc')
```

Enables the display of popup messages at the UE, for ETWS primary notifications.

param network_scope
No help available

return
enable: No help available

set(*network_scope: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:ETWS:POPup
driver.configure.signaling.etws.popup.set(network_scope = 'abc', enable = False)
```

Enables the display of popup messages at the UE, for ETWS primary notifications.

param network_scope
No help available

param enable
No help available

6.3.4.4.4 Secondary

class SecondaryCls

Secondary commands group definition. 7 total commands, 7 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.4.4.4.1 Cgroup

SCPI Command :

```
[CONFigure]:SIGNaling:ETWS:SECondary:CGRoup
```

class CgroupCls

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

get(*network_scope: str*) → CodingGroup

```
# SCPI: [CONFigure]:SIGNaling:ETWS:SECondary:CGRoup
value: enums.CodingGroup = driver.configure.signaling.etws.secondary.cgroup.
↪get(network_scope = 'abc')
```

Selects the coding group for ETWS secondary notifications.

param network_scope
No help available

return
coding_group: G7: GSM 7-bit coding without language string G7L: GSM 7-bit coding with language string U2L: UCS-2 coding with language string

set(*network_scope: str*, *coding_group: CodingGroup*) → None

```
# SCPI: [CONFigure]:SIGNaling:ETWS:SECondary:CGRoup
driver.configure.signaling.etws.secondary.cgroup.set(network_scope = 'abc',␣
↪coding_group = enums.CodingGroup.G7)
```

Selects the coding group for ETWS secondary notifications.

param network_scope
No help available

param coding_group
G7: GSM 7-bit coding without language string G7L: GSM 7-bit coding with language string U2L: UCS-2 coding with language string

6.3.4.4.4.2 Data

SCPI Command :

```
[CONFigure]:SIGNaling:ETWS:SECondary:DATA
```

class DataCls

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

get(*network_scope: str*) → str

```
# SCPI: [CONFigure]:SIGNaling:ETWS:SECondary:DATA
value: str = driver.configure.signaling.etws.secondary.data.get(network_scope =
↪'abc')
```

Defines the broadcasted ETWS secondary notification text.

param network_scope
No help available

return
data: No help available

set(*network_scope: str*, *data: str*) → None

```
# SCPI: [CONFigure]:SIGNaling:ETWS:SECondary:DATA
driver.configure.signaling.etws.secondary.data.set(network_scope = 'abc', data␣
↪= 'abc')
```

Defines the broadcasted ETWS secondary notification text.

param network_scope
No help available

param data
No help available

6.3.4.4.4.3 Id

SCPI Command :

```
[CONFigure]:SIGNaling:ETWS:SECondary:ID
```

class IdCls

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

get(*network_scope: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:ETWS:SECondary:ID
value: int = driver.configure.signaling.etws.secondary.id.get(network_scope =
↪'abc')
```

Defines the message identifier for ETWS secondary notifications.

param network_scope
No help available

return
identifier: No help available

set(*network_scope: str*, *identifier: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:ETWS:SECondary:ID
driver.configure.signaling.etws.secondary.id.set(network_scope = 'abc',␣
↪identifier = 1)
```

Defines the message identifier for ETWS secondary notifications.

param network_scope
No help available

param identifier
No help available

6.3.4.4.4.4 Language

SCPI Command :

```
[CONFigure]:SIGNaling:ETWS:SECondary:LANGuage
```

class LanguageCls

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

class GetStruct

Response structure. Fields:

- Language: enums.LanguageB: Two-character language string
- Cgroup_Language: enums.GroupLanguage: Selects the coding group (GSM 7-bit or UCS-2) . Omitting the value sets G7L.

get(*network_scope: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:ETWS:SECondary:LANGuage
value: GetStruct = driver.configure.signaling.etws.secondary.language.
↪get(network_scope = 'abc')
```

Selects the language string and sets the coding group for ETWS secondary notifications.

param network_scope
No help available

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

set(*network_scope: str*, *language: LanguageB*, *cgroup_language: GroupLanguage = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:ETWS:SECondary:LANGuage
driver.configure.signaling.etws.secondary.language.set(network_scope = 'abc',␣
↪language = enums.LanguageB.DANish, cgroup_language = enums.GroupLanguage.G7L)
```

Selects the language string and sets the coding group for ETWS secondary notifications.

param network_scope
No help available

param language
(enum or string) Two-character language string

param cgroup_language
Selects the coding group (GSM 7-bit or UCS-2) . Omitting the value sets G7L.

6.3.4.4.4.5 Serial

SCPI Command :

```
[CONFigure]:SIGNaling:ETWS:SECondary:SERial
```

class SerialCls

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

class GetStruct

Response structure. Fields:

- Display_Mode: enums.DisplayMode: IMMediate: The UE displays a message immediately. NORMal: The UE displays a message upon user action.
- Message_Code: int: The last 8 bits of the message code of the serial number.
- Update_Time: int: The version of the message transmitted as an update number.

get(*network_scope: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:ETWS:SECondary:SERial
value: GetStruct = driver.configure.signaling.etws.secondary.serial.get(network_
↪scope = 'abc')
```

Defines settings influencing the serial number for ETWS secondary notifications.

param network_scope
No help available

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

set(*network_scope: str*, *display_mode: DisplayMode*, *message_code: int = None*, *update_time: int = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:ETWS:SECondary:SERial
driver.configure.signaling.etws.secondary.serial.set(network_scope = 'abc',␣
↪display_mode = enums.DisplayMode.IMMediate, message_code = 1, update_time = 1)
```

Defines settings influencing the serial number for ETWS secondary notifications.

param network_scope
No help available

param display_mode
IMMediate: The UE displays a message immediately. NORMal: The UE displays a message upon user action.

param message_code
The last 8 bits of the message code of the serial number.

param update_time
The version of the message transmitted as an update number.

6.3.4.4.4.6 Tranmission

SCPI Command :

```
[CONFigure]:SIGNaling:ETWS:SECondary:TRANmission
```

class TranmissionCls

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

get(*network_scope: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:ETWS:SECondary:TRANmission
value: bool = driver.configure.signaling.etws.secondary.tranmission.get(network_
↪scope = 'abc')
```

Enables the transmission of ETWS secondary notifications via the SIB.

param network_scope
No help available

return
enable: No help available

set(*network_scope: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:ETWS:SECondary:TRANmission
driver.configure.signaling.etws.secondary.tranmission.set(network_scope = 'abc',
↪ enable = False)
```

Enables the transmission of ETWS secondary notifications via the SIB.

param network_scope
No help available

param enable
No help available

6.3.4.4.4.7 WoLanguage

SCPI Command :

```
[CONFigure]:SIGNaling:ETWS:SECondary:WOLanguage
```

class WoLanguageCls

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

get(*network_scope: str*) → LanguageB

```
# SCPI: [CONFigure]:SIGNaling:ETWS:SECondary:WOLanguage
value: enums.LanguageB = driver.configure.signaling.etws.secondary.woLanguage.
↪get(network_scope = 'abc')
```

Selects the language and sets the coding group 'GSM 7-bit coding without language string', for ETWS secondary notifications.

param network_scope
No help available

return
language: No help available

set(*network_scope: str*, *language: LanguageB*) → None

```
# SCPI: [CONFigure]:SIGNaling:ETWS:SECondary:WOLanguage
driver.configure.signaling.etws.secondary.woLanguage.set(network_scope = 'abc',␣
↪language = enums.LanguageB.DANish)
```

Selects the language and sets the coding group 'GSM 7-bit coding without language string', for ETWS secondary notifications.

param network_scope
No help available

param language
No help available

6.3.4.4.5 Serial

SCPI Command :

```
[CONFigure]:SIGNaling:ETWS:SERial
```

class SerialCls

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

class GetStruct

Response structure. Fields:

- Display_Mode: enums.DisplayMode: IMMediate: The UE displays a message immediately. NORMal: The UE displays a message upon user action.
- Message_Code: int: The last 8 bits of the message code of the serial number.
- Update_Time: int: The version of the message transmitted as an update number.

get(*network_scope: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:ETWS:SERial
value: GetStruct = driver.configure.signaling.etws.serial.get(network_scope =
↪'abc')
```

Defines settings influencing the serial number for ETWS primary notifications.

param network_scope
No help available

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

set(*network_scope: str*, *display_mode: DisplayMode*, *message_code: int = None*, *update_time: int = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:ETWS:SERial
driver.configure.signaling.etws.serial.set(network_scope = 'abc', display_mode
↪= enums.DisplayMode.IMMediate, message_code = 1, update_time = 1)
```

Defines settings influencing the serial number for ETWS primary notifications.

param network_scope
No help available

param display_mode
IMMediate: The UE displays a message immediately. NORMal: The UE displays a message upon user action.

param message_code
The last 8 bits of the message code of the serial number.

param update_time
The version of the message transmitted as an update number.

6.3.4.4.6 Tranmission

SCPI Command :

```
[CONFigure]:SIGNaling:ETWS:TRANmission
```

class TranmissionCls

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

get(*network_scope: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:ETWS:TRANmission
value: bool = driver.configure.signaling.etws.tranmission.get(network_scope =
↪'abc')
```

Enables the transmission of ETWS primary notifications via the SIB.

param network_scope
No help available

return
enable: No help available

set(*network_scope: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:ETWS:TRANmission
driver.configure.signaling.etws.tranmission.set(network_scope = 'abc', enable =
↪False)
```

Enables the transmission of ETWS primary notifications via the SIB.

param network_scope
No help available

param enable
No help available

6.3.4.5 Fading

class FadingCls

Fading commands group definition. 7 total commands, 5 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.4.5.1 Dshift

SCPI Command :

```
[CONFigure]:SIGNaling:FADing:DSHift
```

class DshiftCls

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

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:FADing:DSHift
value: float = driver.configure.signaling.fading.dshift.get(cell_name = 'abc')
```

Sets the maximum Doppler frequency for fading. A setting is only allowed in USER mode (see [CONFigure:]SIGNaling:FADing:DSHift:MODE) .

param cell_name
No help available

return
frequency: No help available

set(*cell_name: str*, *frequency: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:FADing:DSHift
driver.configure.signaling.fading.dshift.set(cell_name = 'abc', frequency = 1.0)
```

Sets the maximum Doppler frequency for fading. A setting is only allowed in USER mode (see [CONFigure:]SIGNaling:FADing:DSHift:MODE) .

param cell_name
No help available

param frequency
No help available

Cloning the Group

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

Subgroups

6.3.4.5.1.1 Mode

SCPI Command :

```
[CONFigure]:SIGNaling:FADing:DSHift:MODE
```

class ModeCls

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

get(*cell_name: str*) → FadingMode

```
# SCPI: [CONFigure]:SIGNaling:FADing:DSHift:MODE
value: enums.FadingMode = driver.configure.signaling.fading.dshift.mode.
↪get(cell_name = 'abc')
```

Selects the Doppler shift mode for fading.

param cell_name
No help available

return
mode: NORMal: The maximum Doppler frequency is determined by the fading profile. USER: The maximum Doppler frequency is configurable.

set(*cell_name: str*, *mode: FadingMode*) → None

```
# SCPI: [CONFigure]:SIGNaling:FADing:DSHift:MODE
driver.configure.signaling.fading.dshift.mode.set(cell_name = 'abc', mode =␣
↪enums.FadingMode.NORMal)
```

Selects the Doppler shift mode for fading.

param cell_name
No help available

param mode
NORMal: The maximum Doppler frequency is determined by the fading profile. USER: The maximum Doppler frequency is configurable.

6.3.4.5.2 Enable

SCPI Command :

```
[CONFigure]:SIGNaling:FADing:ENABle
```

class EnableCls

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

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:FADing:ENABle
value: bool = driver.configure.signaling.fading.enable.get(cell_name = 'abc')
```

Enables/disables fading for a cell.

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:FADing:ENABle
driver.configure.signaling.fading.enable.set(cell_name = 'abc', enable = False)
```

Enables/disables fading for a cell.

param cell_name
No help available

param enable
No help available

6.3.4.5.3 Iloss

SCPI Command :

```
[CONFigure]:SIGNaling:FADing:ILOSs
```

class IlossCls

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

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:FADing:ILOSs
value: float = driver.configure.signaling.fading.iloss.get(cell_name = 'abc')
```

Sets the insertion loss for fading.

param cell_name
No help available

return
insertion_loss: No help available

set(*cell_name: str*, *insertion_loss: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:FADing:ILOSs
driver.configure.signaling.fading.iloss.set(cell_name = 'abc', insertion_loss =␣
↪1.0)
```

Sets the insertion loss for fading.

param cell_name
No help available

param insertion_loss
No help available

Cloning the Group

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

Subgroups

6.3.4.5.3.1 Mode

SCPI Command :

```
[CONFigure]:SIGNaling:FADing:ILOSs:MODE
```

class ModeCls

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

get(*cell_name: str*) → FadingMode

```
# SCPI: [CONFigure]:SIGNaling:FADing:ILOSs:MODE
value: enums.FadingMode = driver.configure.signaling.fading.iloss.mode.get(cell_
↪name = 'abc')
```

No command help available

param cell_name
No help available

return
mode: No help available

set(*cell_name: str*, *mode: FadingMode*) → None

```
# SCPI: [CONFigure]:SIGNaling:FADing:ILOSs:MODE
driver.configure.signaling.fading.iloss.mode.set(cell_name = 'abc', mode =␣
↪enums.FadingMode.NORMal)
```

No command help available

param cell_name
No help available

param mode
No help available

6.3.4.5.4 Profile

SCPI Command :

```
[CONFigure]:SIGNaling:FADing:PROFile
```

class ProfileCls

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

get(*cell_name: str*) → FadingProfile

```
# SCPI: [CONFigure]:SIGNaling:FADing:PROFile
value: enums.FadingProfile = driver.configure.signaling.fading.profile.get(cell_
↪name = 'abc')
```

Selects a propagation condition profile for fading.

param cell_name
No help available

return
profile: NONE and values listed in Table 'Profile values'

set(*cell_name: str*, *profile: FadingProfile*) → None

```
# SCPI: [CONFigure]:SIGNaling:FADing:PROFile
driver.configure.signaling.fading.profile.set(cell_name = 'abc', profile =␣
↪enums.FadingProfile.CTES)
```

Selects a propagation condition profile for fading.

param cell_name
No help available

param profile
NONE and values listed in Table 'Profile values'

6.3.4.5.5 Seed

SCPI Command :

```
[CONFigure]:SIGNaling:FADing:SEED
```

class SeedCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:FADing:SEED
value: int = driver.configure.signaling.fading.seed.get(cell_name = 'abc')
```

Sets the start seed for the pseudo-random fading algorithm.

param cell_name
No help available

return
start_seed: No help available

set(*cell_name: str*, *start_seed: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:FADing:SEED
driver.configure.signaling.fading.seed.set(cell_name = 'abc', start_seed = 1)
```

Sets the start seed for the pseudo-random fading algorithm.

param cell_name
No help available

param start_seed
No help available

6.3.4.6 Fgs

SCPI Command :

```
[CONFigure]:SIGNaling:FGS:TMODe
```

class FgsCls

Fgs commands group definition. 27 total commands, 6 Subgroups, 1 group commands

get_tmode() → bool

```
# SCPI: [CONFigure]:SIGNaling:FGS:TMODe
value: bool = driver.configure.signaling.fgs.get_tmode()
```

Selects whether an 'ACTIVATE TEST MODE' message is sent to the UE during registration in a 5GS tracking area, or not. Configure this setting before registration.

return
enable: No help available

set_tmode(*enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:FGS:TMODe
driver.configure.signaling.fgs.set_tmode(enable = False)
```

Selects whether an 'ACTIVATE TEST MODE' message is sent to the UE during registration in a 5GS tracking area, or not. Configure this setting before registration.

param enable
No help available

Cloning the Group

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

Subgroups

6.3.4.6.1 AsPy

class AsPyCls

AsPy commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.4.6.1.1 Security

SCPI Commands :

```
[CONFigure]:SIGNaling:FGS:AS:SECurity:INTegrity
[CONFigure]:SIGNaling:FGS:AS:SECurity:CIPHering
```

class SecurityCls

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

get_ciphering() → SecurityAlgorithm

```
# SCPI: [CONFigure]:SIGNaling:FGS:AS:SECurity:CIPHering
value: enums.SecurityAlgorithm = driver.configure.signaling.fgs.asPy.security.
↪get_ciphering()
```

Selects an algorithm for AS ciphering in 5GS tracking areas.

return
algorithm: No help available

get_integrity() → SecurityAlgorithm

```
# SCPI: [CONFigure]:SIGNaling:FGS:AS:SECurity:INTegrity
value: enums.SecurityAlgorithm = driver.configure.signaling.fgs.asPy.security.
↪get_integrity()
```

Selects an algorithm for AS integrity protection in 5GS tracking areas.

return
algorithm: No help available

set_ciphering(*algorithm: SecurityAlgorithm*) → None

```
# SCPI: [CONFigure]:SIGNaling:FGS:AS:SECurity:CIPHering
driver.configure.signaling.fgs.asPy.security.set_ciphering(algorithm = enums.
↪SecurityAlgorithm.AES)
```

Selects an algorithm for AS ciphering in 5GS tracking areas.

param algorithm
No help available

set_integrity(*algorithm: SecurityAlgorithm*) → None

```
# SCPI: [CONFigure]:SIGNaling:FGS:AS:SECurity:INTegrity
driver.configure.signaling.fgs.asPy.security.set_integrity(algorithm = enums.
↪SecurityAlgorithm.AES)
```

Selects an algorithm for AS integrity protection in 5GS tracking areas.

param algorithm
No help available

6.3.4.6.2 CnPaging

class CnPagingCls

CnPaging commands group definition. 3 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.4.6.2.1 EdRx

SCPI Commands :

```
[CONFigure]:SIGNaling:FGS:CNPaging:EDRX:MODE
[CONFigure]:SIGNaling:FGS:CNPaging:EDRX:EPTWindow
[CONFigure]:SIGNaling:FGS:CNPaging:EDRX:CYCLe
```

class EdRxCls

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

get_cycle() → float

```
# SCPI: [CONFigure]:SIGNaling:FGS:CNPaging:EDRX:CYCLe
value: float = driver.configure.signaling.fgs.cnPaging.edRx.get_cycle()
```

Configures the eDRX cycle length for the mode USER.

return
cycle_length: No help available

get_ept_window() → float

```
# SCPI: [CONFigure]:SIGNaling:FGS:CNPaging:EDRX:EPTWindow
value: float = driver.configure.signaling.fgs.cnPaging.edRx.get_ept_window()
```

Configures the paging time window for the mode USER.

return
time_window: No help available

get_mode() → EdRxMode

```
# SCPI: [CONFigure]:SIGNaling:FGS:CNPaging:EDRX:MODE
value: enums.EdRxMode = driver.configure.signaling.fgs.cnPaging.edRx.get_mode()
```

Selects a mode for configuration of the paging time window and of the cycle length.

return
mode: UERequested: as requested by UE USER: configured values

set_cycle(*cycle_length: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:FGS:CNPaging:EDRX:CYCLe
driver.configure.signaling.fgs.cnPaging.edRx.set_cycle(cycle_length = 1.0)
```

Configures the eDRX cycle length for the mode USER.

param cycle_length
No help available

set_ept_window(*time_window: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:FGS:CNPaging:EDRX:EPTWindow
driver.configure.signaling.fgs.cnPaging.edRx.set_ept_window(time_window = 1.0)
```

Configures the paging time window for the mode USER.

param time_window
No help available

set_mode(*mode: EdRxMode*) → None

```
# SCPI: [CONFigure]:SIGNaling:FGS:CNPaging:EDRX:MODE
driver.configure.signaling.fgs.cnPaging.edRx.set_mode(mode = enums.EdRxMode.
↪UERequested)
```

Selects a mode for configuration of the paging time window and of the cycle length.

param mode
UERequested: as requested by UE USER: configured values

6.3.4.6.3 Dbearer

SCPI Commands :

```
[CONFigure]:SIGNaling:FGS:DBEarer:DNName
[CONFigure]:SIGNaling:FGS:DBEarer:RLCMode
```

class DbearerCls

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

get_dn_name() → str

```
# SCPI: [CONFigure]:SIGNaling:FGS:DBEarer:DNName
value: str = driver.configure.signaling.fgs.dbearer.get_dn_name()
```

Configures the default data network name (DNN) for default flows in 5GS tracking areas.

return
network_name: No help available

get_rlc_mode() → RlcMode

```
# SCPI: [CONFigure]:SIGNaling:FGS:DBEarer:RLCMode
value: enums.RlcMode = driver.configure.signaling.fgs.dbearer.get_rlc_mode()
```

Configures the RLC mode for default flows in 5GS tracking areas.

return
rlc_mode: RLC mode ACK: acknowledged UACK: unacknowledged

set_dn_name(*network_name: str*) → None

```
# SCPI: [CONFigure]:SIGNaling:FGS:DBEarer:DNName
driver.configure.signaling.fgs.dbearer.set_dn_name(network_name = 'abc')
```

Configures the default data network name (DNN) for default flows in 5GS tracking areas.

param network_name
No help available

set_rlc_mode(*rlc_mode: RlcMode*) → None

```
# SCPI: [CONFigure]:SIGNaling:FGS:DBEarer:RLCMode
driver.configure.signaling.fgs.dbearer.set_rlc_mode(rlc_mode = enums.RlcMode.
↪ACK)
```

Configures the RLC mode for default flows in 5GS tracking areas.

param rlc_mode
RLC mode ACK: acknowledged UACK: unacknowledged

6.3.4.6.4 Nas

class NasCls

Nas commands group definition. 10 total commands, 3 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.4.6.4.1 Auth

SCPI Commands :

```
[CONFigure]:SIGNaling:FGS:NAS:AUTH:ENABle
[CONFigure]:SIGNaling:FGS:NAS:AUTH:RAND
[CONFigure]:SIGNaling:FGS:NAS:AUTH:IRES
```

class AuthCls

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

get_enable() → bool

```
# SCPI: [CONFigure]:SIGNaling:FGS:NAS:AUTH:ENABle
value: bool = driver.configure.signaling.fgs.nas.auth.get_enable()
```

Enables authentication for 5GS tracking areas.

return
enable: No help available

get_ires() → bool

```
# SCPI: [CONFigure]:SIGNaling:FGS:NAS:AUTH:IRES
value: bool = driver.configure.signaling.fgs.nas.auth.get_ires()
```

Enables ignoring the RES* in 5GS tracking areas (successful authentication, even if the UE returns a wrong RES* value) .

return
enable: No help available

get_rand() → str

```
# SCPI: [CONFigure]:SIGNaling:FGS:NAS:AUTH:RAND
value: str = driver.configure.signaling.fgs.nas.auth.get_rand()
```

Defines the random number (RAND) to be used for authentication in 5GS tracking areas.

return
rand: No help available

set_enable(*enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:FGS:NAS:AUTH:ENABle
driver.configure.signaling.fgs.nas.auth.set_enable(enable = False)
```

Enables authentication for 5GS tracking areas.

param enable
No help available

set_ires(*enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:FGS:NAS:AUTH:IRES
driver.configure.signaling.fgs.nas.auth.set_ires(enable = False)
```

Enables ignoring the RES* in 5GS tracking areas (successful authentication, even if the UE returns a wrong RES* value) .

param enable
No help available

set_rand(*rand: str*) → None

```
# SCPI: [CONFigure]:SIGNaling:FGS:NAS:AUTH:RAND
driver.configure.signaling.fgs.nas.auth.set_rand(rand = 'abc')
```

Defines the random number (RAND) to be used for authentication in 5GS tracking areas.

param rand
No help available

6.3.4.6.4.2 Security

SCPI Commands :

```
[CONFigure]:SIGNaling:FGS:NAS:SECurity:ENABle
[CONFigure]:SIGNaling:FGS:NAS:SECurity:INTegrity
[CONFigure]:SIGNaling:FGS:NAS:SECurity:CIPHering
[CONFigure]:SIGNaling:FGS:NAS:SECurity:PAUTh
[CONFigure]:SIGNaling:FGS:NAS:SECurity:PSAuth
```

class SecurityCls

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

get_ciphering() → CipherAlgorithm

```
# SCPI: [CONFigure]:SIGNaling:FGS:NAS:SECurity:CIPHering
value: enums.CipherAlgorithm = driver.configure.signaling.fgs.nas.security.get_
↪ciphering()
```

Selects an algorithm for NAS ciphering in 5GS tracking areas.

return
algorithm: No help available

get_enable() → bool

```
# SCPI: [CONFigure]:SIGNaling:FGS:NAS:SECurity:ENABle
value: bool = driver.configure.signaling.fgs.nas.security.get_enable()
```

Enables security procedures (ciphering, integrity protection) for 5GS tracking areas.

return
enable: No help available

get_integrity() → IntegrityAlgorithm

```
# SCPI: [CONFigure]:SIGNaling:FGS:NAS:SECurity:INTegrity
value: enums.IntegrityAlgorithm = driver.configure.signaling.fgs.nas.security.
↪get_integrity()
```

Selects an algorithm for NAS integrity protection in 5GS tracking areas.

return
algorithm: No help available

get_pauth() → AuthProcedure

```
# SCPI: [CONFigure]:SIGNaling:FGS:NAS:SECurity:PAUTh
value: enums.AuthProcedure = driver.configure.signaling.fgs.nas.security.get_
↪pauth()
```

Selects a primary authentication and key agreement procedure for 5GS tracking areas.

return
procedure: No help available

get_ps_auth() → bool

```
# SCPI: [CONFigure]:SIGNaling:FGS:NAS:SECurity:PSAuth
value: bool = driver.configure.signaling.fgs.nas.security.get_ps_auth()
```

Enables authentication for PDU session establishment.

return
enable: No help available

set_ciphering(*algorithm: CipherAlgorithm*) → None

```
# SCPI: [CONFigure]:SIGNaling:FGS:NAS:SECurity:CIPHering
driver.configure.signaling.fgs.nas.security.set_ciphering(algorithm = enums.
↪CipherAlgorithm.EA0)
```

Selects an algorithm for NAS ciphering in 5GS tracking areas.

param algorithm
No help available

set_enable(*enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:FGS:NAS:SECurity:ENABle
driver.configure.signaling.fgs.nas.security.set_enable(enable = False)
```

Enables security procedures (ciphering, integrity protection) for 5GS tracking areas.

param enable
No help available

set_integrity(*algorithm: IntegrityAlgorithm*) → None

```
# SCPI: [CONFigure]:SIGNaling:FGS:NAS:SECurity:INTegrity
driver.configure.signaling.fgs.nas.security.set_integrity(algorithm = enums.
↪IntegrityAlgorithm.HIGHest)
```

Selects an algorithm for NAS integrity protection in 5GS tracking areas.

param algorithm
No help available

set_pauth(*procedure: AuthProcedure*) → None

```
# SCPI: [CONFigure]:SIGNaling:FGS:NAS:SECurity:PAUTh
driver.configure.signaling.fgs.nas.security.set_pauth(procedure = enums.
↪AuthProcedure.EAKA)
```

Selects a primary authentication and key agreement procedure for 5GS tracking areas.

param procedure
No help available

set_ps_auth(*enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:FGS:NAS:SECurity:PSAuth
driver.configure.signaling.fgs.nas.security.set_ps_auth(enable = False)
```

Enables authentication for PDU session establishment.

param enable
No help available

6.3.4.6.4.3 Tlv

SCPI Commands :

```
[CONFigure]:SIGNaling:FGS:NAS:TLV:REGaccept
[CONFigure]:SIGNaling:FGS:NAS:TLV:PDUaccept
```

class TlvCls

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

set_pdu_accept(*message: str*) → None

```
# SCPI: [CONFigure]:SIGNaling:FGS:NAS:TLV:PDUaccept
driver.configure.signaling.fgs.nas.tlv.set_pdu_accept(message = 'abc')
```

No command help available

param message
No help available

set_reg_accept(*message: str*) → None

```
# SCPI: [CONFigure]:SIGNaling:FGS:NAS:TLV:REGaccept
driver.configure.signaling.fgs.nas.tlv.set_reg_accept(message = 'abc')
```

No command help available

param message
No help available

6.3.4.6.5 Nbehavior

SCPI Commands :

```
[CONFigure]:SIGNaling:FGS:NBEHavior:DITimer
[CONFigure]:SIGNaling:FGS:NBEHavior:KRRC
```

class NbehaviorCls

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

get_di_timer() → float

```
# SCPI: [CONFigure]:SIGNaling:FGS:NBEHavior:DITimer
value: float or bool = driver.configure.signaling.fgs.nbehavior.get_di_timer()
```

Configures the data inactivity timer for 5GS tracking areas. With enabled timer, an RRC connection is released when there has been no activity on the connection (no traffic) for the configured time.

return
timer: (float or boolean) Numeric value: Enables the timer and sets the timer value. ON: Enables the timer, using the configured numeric value. OFF: Disables the timer (no release due to inactivity) .

get_krrc() → bool

```
# SCPI: [CONFigure]:SIGNaling:FGS:NBEHavior:KRRC
value: bool = driver.configure.signaling.fgs.nbehavior.get_krrc()
```

Selects whether the RRC connection is kept after a registration in a 5GS tracking area.

return
enable: No help available

set_di_timer(*timer: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:FGS:NBEHavior:DITimer
driver.configure.signaling.fgs.nbehavior.set_di_timer(timer = 1.0)
```

Configures the data inactivity timer for 5GS tracking areas. With enabled timer, an RRC connection is released when there has been no activity on the connection (no traffic) for the configured time.

param timer
(float or boolean) Numeric value: Enables the timer and sets the timer value. ON: Enables the timer, using the configured numeric value. OFF: Disables the timer (no release due to inactivity) .

set_krrc(*enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:FGS:NBEHavior:KRRC
driver.configure.signaling.fgs.nbehavior.set_krrc(enable = False)
```

Selects whether the RRC connection is kept after a registration in a 5GS tracking area.

param enable
No help available

Cloning the Group

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

Subgroups

6.3.4.6.5.1 RrcReject

SCPI Command :

```
[CONFigure]:SIGNaling:FGS:NBEHavior:RRCReject
```

class RrcRejectCls

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

class RrcRejectStruct

Response structure. Fields:

- Reject_Procedure: enums.FgsRejectProcedure: Selects the procedure to fail. NOR: No reject. REGR: 'Register Reject' message AUTR: 'Authentication Reject' message PDUR: 'PDU Session Establishment Reject' message
- Reject_Cause: enums.FgsRejectCause: Selects the cause to be signaled to the UE as the reason for the rejection.

get() → RrcRejectStruct

```
# SCPI: [CONFigure]:SIGNaling:FGS:NBEHavior:RRCReject
value: RrcRejectStruct = driver.configure.signaling.fgs.nbehavior.rrcReject.
↪get()
```

Configures RRC reject causes for 5GS tracking areas.

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

set(*reject_procedure: FgsRejectProcedure*, *reject_cause: FgsRejectCause = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:FGS:NBEHavior:RRCReject
driver.configure.signaling.fgs.nbehavior.rrcReject.set(reject_procedure = enums.
↪FgsRejectProcedure.AUTR, reject_cause = enums.FgsRejectCause.C003)
```

Configures RRC reject causes for 5GS tracking areas.

param reject_procedure
Selects the procedure to fail. NOR: No reject. REGR: 'Register Reject' message AUTR: 'Authentication Reject' message PDUR: 'PDU Session Establishment Reject' message

param reject_cause
Selects the cause to be signaled to the UE as the reason for the rejection.

6.3.4.6.6 UeCapability

SCPI Commands :

```
[CONFigure]:SIGNaling:FGS:UECapability:MODE
[CONFigure]:SIGNaling:FGS:UECapability:SEGMentation
[CONFigure]:SIGNaling:FGS:UECapability:SRSCarrier
```

class UeCapabilityCls

UeCapability commands group definition. 6 total commands, 2 Subgroups, 3 group commands

get_mode() → ModeUeCapability

```
# SCPI: [CONFigure]:SIGNaling:FGS:UECapability:MODE
value: enums.ModeUeCapability = driver.configure.signaling.fgs.ueCapability.get_
↪mode()
```

Selects the configuration mode for 'UeCapabilityEnquiry' messages in 5GS tracking areas.

return
mode: SKIP: no 'UeCapabilityEnquiry' messages AUTO: automatic message configuration UDEFined: configuration via the other commands in this chapter

get_segmentation() → bool

```
# SCPI: [CONFigure]:SIGNaling:FGS:UECapability:SEGMentation
value: bool = driver.configure.signaling.fgs.ueCapability.get_segmentation()
```

Selects whether the UE is allowed to use segmentation for capability information in 5GS tracking areas.

return
enable: No help available

get_srs_carrier() → bool

```
# SCPI: [CONFigure]:SIGNaling:FGS:UECapability:SRSCarrier
value: bool = driver.configure.signaling.fgs.ueCapability.get_srs_carrier()
```

Selects whether the UE must report capabilities for SRS carrier switching. The flag is sent as 'srsSwitchingTimeRequest'.

return
srs_enable: No help available

set_mode(*mode: ModeUeCapability*) → None

```
# SCPI: [CONFigure]:SIGNaling:FGS:UECapability:MODE
driver.configure.signaling.fgs.ueCapability.set_mode(mode = enums.
↪ModeUeCapability.AUTO)
```

Selects the configuration mode for 'UeCapabilityEnquiry' messages in 5GS tracking areas.

param mode
SKIP: no 'UeCapabilityEnquiry' messages AUTO: automatic message configuration UDEFined: configuration via the other commands in this chapter

set_segmentation(*enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:FGS:UECapability:SEGMentation
driver.configure.signaling.fgs.ueCapability.set_segmentation(enable = False)
```

Selects whether the UE is allowed to use segmentation for capability information in 5GS tracking areas.

param enable
No help available

set_srs_carrier(*srs_enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:FGS:UECapability:SRSCarrier
driver.configure.signaling.fgs.ueCapability.set_srs_carrier(srs_enable = False)
```

Selects whether the UE must report capabilities for SRS carrier switching. The flag is sent as 'srsSwitchingTimeRequest'.

param srs_enable
No help available

Cloning the Group

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

Subgroups

6.3.4.6.6.1 Eutra

SCPI Commands :

```
[CONFigure]:SIGNaling:FGS:UECapability:EUTRa:RFORmat
[CONFigure]:SIGNaling:FGS:UECapability:EUTRa:SFC
```

class EutraCls

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

get_rformat() → bool

```
# SCPI: [CONFigure]:SIGNaling:FGS:UECapability:EUTRa:RFORmat
value: bool = driver.configure.signaling.fgs.ueCapability.eutra.get_rformat()
```

Adds the field 'requestReducedFormat-r13' to the message 'UeCapabilityEnquiry', for 5GS tracking areas, container type 'UE-EUTRA-Capability'.

return
enable: No help available

get_sfc() → bool

```
# SCPI: [CONFigure]:SIGNaling:FGS:UECapability:EUTRa:SFC
value: bool = driver.configure.signaling.fgs.ueCapability.eutra.get_sfc()
```

Adds the field 'requestSkipFallbackComb-r13' to the message 'UeCapabilityEnquiry', for 5GS tracking areas, container type 'UE-EUTRA-Capability'.

return
enable: No help available

set_rformat(*enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:FGS:UECapability:EUTRa:RFORmat
driver.configure.signaling.fgs.ueCapability.eutra.set_rformat(enable = False)
```

Adds the field 'requestReducedFormat-r13' to the message 'UeCapabilityEnquiry', for 5GS tracking areas, container type 'UE-EUTRA-Capability'.

param enable
No help available

set_sfc(*enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:FGS:UECapability:EUTRa:SFC
driver.configure.signaling.fgs.ueCapability.eutra.set_sfc(enable = False)
```

Adds the field 'requestSkipFallbackComb-r13' to the message 'UeCapabilityEnquiry', for 5GS tracking areas, container type 'UE-EUTRA-Capability'.

param enable
No help available

6.3.4.6.6.2 Mrdc

SCPI Command :

```
[CONFigure]:SIGNaling:FGS:UECapability:MRDC:ENRonly
```

class MrdcCls

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

get_enr_only() → bool

```
# SCPI: [CONFigure]:SIGNaling:FGS:UECapability:MRDC:ENRonly
value: bool = driver.configure.signaling.fgs.ueCapability.mrdc.get_enr_only()
```

Adds the field 'eutra-nr-only-r15' to the message 'UeCapabilityEnquiry', for 5GS tracking areas, container type 'UE-MRDC-Capability'.

return
enable: No help available

set_enr_only(*enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:FGS:UECapability:MRDC:ENRonly
driver.configure.signaling.fgs.ueCapability.mrdc.set_enr_only(enable = False)
```

Adds the field 'eutra-nr-only-r15' to the message 'UeCapabilityEnquiry', for 5GS tracking areas, container type 'UE-MRDC-Capability'.

param enable
No help available

6.3.4.7 Lte

class LteCls

Lte commands group definition. 221 total commands, 4 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.4.7.1 Ca

class CaCls

Ca commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.4.7.1.1 Scell

class ScellCls

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

Cloning the Group

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

Subgroups

6.3.4.7.1.2 Activation

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CA:SCELl:ACTivation
```

class ActivationCls

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

get(*cell_name: List[str]*) → List[bool]

```
# SCPI: [CONFigure]:SIGNaling:LTE:CA:SCELl:ACTivation
value: List[bool] = driver.configure.signaling.lte.ca.scell.activation.get(cell_
↪name = ['abc1', 'abc2', 'abc3'])
```

Triggers the manual MAC activation or MAC deactivation for an SCell or several SCells. A query returns the current MAC activation state for an SCell.

param cell_name
No help available

return
activation: ON: Activate MAC (setting) / MAC is active (query) . OFF: Deactivate MAC (setting) / MAC is inactive (query) .

set(*cell_name: List[str]*, *activation: List[bool]*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CA:SCELl:ACTivation
driver.configure.signaling.lte.ca.scell.activation.set(cell_name = ['abc1',
↪'abc2', 'abc3'], activation = [True, False, True])
```

Triggers the manual MAC activation or MAC deactivation for an SCell or several SCells. A query returns the current MAC activation state for an SCell.

param cell_name
No help available

param activation
ON: Activate MAC (setting) / MAC is active (query) . OFF: Deactivate MAC (setting) / MAC is inactive (query) .

6.3.4.7.1.3 Uplink

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CA:SCELl:UL
```

class UplinkCls

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

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:LTE:CA:SCELl:UL
value: bool = driver.configure.signaling.lte.ca.scell.uplink.get(cell_name =
↪'abc')
```

Enables or disables the uplink of an SCell.

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CA:SCELl:UL
driver.configure.signaling.lte.ca.scell.uplink.set(cell_name = 'abc', enable =␣
↪False)
```

Enables or disables the uplink of an SCell.

param cell_name
No help available

param enable
No help available

6.3.4.7.2 Cell

class CellCls

Cell commands group definition. 214 total commands, 23 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.4.7.2.1 Antenna

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:ANTenna
```

class AntennaCls

Antenna commands group definition. 4 total commands, 3 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Ant_No_Ports: enums.AntNoPorts: Number of CRS antenna ports.
- Beam_No_Ports: enums.BeamNoPorts: Beamforming number of antenna ports.
- Dl_Iq_Data_Streams: enums.DlIqDataStreams: Number of I/Q data DL streams.

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:ANTenna
value: GetStruct = driver.configure.signaling.lte.cell.antenna.get(cell_name =
↪'abc')
```

Selects the number of antenna ports and streams.

param cell_name
No help available

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

set(*cell_name: str*, *ant_no_ports: AntNoPorts*, *beam_no_ports: BeamNoPorts = None*, *dl_iq_data_streams: DlIqDataStreams = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:ANTenna
driver.configure.signaling.lte.cell.antenna.set(cell_name = 'abc', ant_no_ports␣
↪= enums.AntNoPorts.P1, beam_no_ports = enums.BeamNoPorts.NONE, dl_iq_data_
↪streams = enums.DlIqDataStreams.S1)
```

Selects the number of antenna ports and streams.

param cell_name
No help available

param ant_no_ports
Number of CRS antenna ports.

param beam_no_ports
Beamforming number of antenna ports.

param dl_iq_data_streams
Number of I/Q data DL streams.

Cloning the Group

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

Subgroups

6.3.4.7.2.2 Beamforming

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:ANTenna:BEAMforming
```

class BeamformingCls

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

get(*cell_name: str*) → BeamNoPorts

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:ANTenna:BEAMforming
value: enums.BeamNoPorts = driver.configure.signaling.lte.cell.antenna.
↪beamforming.get(cell_name = 'abc')
```

Sets the number of beamforming antenna ports.

param cell_name
No help available

return
beam_no_ports: No help available

set(*cell_name: str*, *beam_no_ports: BeamNoPorts*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:ANTenna:BEAMforming
driver.configure.signaling.lte.cell.antenna.beamforming.set(cell_name = 'abc',␣
↪beam_no_ports = enums.BeamNoPorts.NONE)
```

Sets the number of beamforming antenna ports.

param cell_name
No help available

param beam_no_ports
No help available

6.3.4.7.2.3 CrSports

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:ANTenna:CRSPorts
```

class CrSportsCls

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

get(*cell_name: str*) → AntNoPorts

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:ANTenna:CRSPorts
value: enums.AntNoPorts = driver.configure.signaling.lte.cell.antenna.crSports.
↪get(cell_name = 'abc')
```

Sets the number of CRS antenna ports.

param cell_name
No help available

return
ant_no_ports: No help available

set(*cell_name: str*, *ant_no_ports: AntNoPorts*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:ANTenna:CRSPorts
driver.configure.signaling.lte.cell.antenna.crSports.set(cell_name = 'abc', ant_
↪no_ports = enums.AntNoPorts.P1)
```

Sets the number of CRS antenna ports.

param cell_name
No help available

param ant_no_ports
No help available

6.3.4.7.2.4 Streams

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:ANTenna:STReams
```

class StreamsCls

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

get(*cell_name: str*) → DlIqDataStreams

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:ANTenna:STReams
value: enums.DlIqDataStreams = driver.configure.signaling.lte.cell.antenna.
↪streams.get(cell_name = 'abc')
```

Sets the number of I/Q data streams.

param cell_name
No help available

return
dl_iq_data_streams: No help available

set(*cell_name: str*, *dl_iq_data_streams: DlIqDataStreams*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:ANTenna:STReams
driver.configure.signaling.lte.cell.antenna.streams.set(cell_name = 'abc', dl_
↪iq_data_streams = enums.DlIqDataStreams.S1)
```

Sets the number of I/Q data streams.

param cell_name
No help available

param dl_iq_data_streams
No help available

6.3.4.7.2.5 Barred

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:BARRed
```

class BarredCls

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

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:BARRed
value: bool = driver.configure.signaling.lte.cell.barred.get(cell_name = 'abc')
```

Specifies whether the cell is barred.

param cell_name
No help available

return
enable: ON: cell barred OFF: cell not barred

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:BARRed
driver.configure.signaling.lte.cell.barred.set(cell_name = 'abc', enable =␣
↪False)
```

Specifies whether the cell is barred.

param cell_name
No help available

param enable
ON: cell barred OFF: cell not barred

6.3.4.7.2.6 BbCombining

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:BBCombining
```

class BbCombiningCls

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

get(*cell_name: str*) → bool

```
# SCPI: [CONfigure]:SIGNaling:LTE:CELL:BBCombining
value: bool = driver.configure.signaling.lte.cell.bbCombining.get(cell_name =
↪'abc')
```

Allows baseband combining for the cell.

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONfigure]:SIGNaling:LTE:CELL:BBCombining
driver.configure.signaling.lte.cell.bbCombining.set(cell_name = 'abc', enable =␣
↪False)
```

Allows baseband combining for the cell.

param cell_name
No help available

param enable
No help available

6.3.4.7.2.7 Cdrx

class CdrxCls

Cdrx commands group definition. 10 total commands, 7 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.4.7.2.8 AaScheduler

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:CDRX:AASCheduler
```

class AaSchedulerCls

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

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:CDRX:AASCheduler
value: bool = driver.configure.signaling.lte.cell.cdrx.aaScheduler.get(cell_
↪name = 'abc')
```

Enables or disables automatic scheduling to ensure gaps for DRX opportunities. If connected DRX is disabled, this setting is ignored (implicit OFF) . Enable connected DRX via [CONFigure:]SIGNaling:LTE:CELL:CDRX:ENABle.

param cell_name
No help available

return
enable: - ON: The scheduler allocates DL resources only if there is DL data. Without queued DL data, no DL resources are allocated and there is an opportunity for DRX. In the UL direction, the scheduler allocates resources only upon request by the UE. - OFF: The configured scheduling applies.

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:CDRX:AASCheduler
driver.configure.signaling.lte.cell.cdrx.aaScheduler.set(cell_name = 'abc',␣
↪enable = False)
```

Enables or disables automatic scheduling to ensure gaps for DRX opportunities. If connected DRX is disabled, this setting is ignored (implicit OFF) . Enable connected DRX via [CONFigure:]SIGNaling:LTE:CELL:CDRX:ENABle.

param cell_name
No help available

param enable

- ON: The scheduler allocates DL resources only if there is DL data. Without queued DL data, no DL resources are allocated and there is an opportunity for DRX. In the UL direction, the scheduler allocates resources only upon request by the UE.
- OFF: The configured scheduling applies.

6.3.4.7.2.9 Enable

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:CDRX:ENABle
```

class EnableCls

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

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:CDRX:ENABle
value: bool = driver.configure.signaling.lte.cell.cdrx.enable.get(cell_name =
↪'abc')
```

Enables or disables connected DRX.

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:CDRX:ENABle
driver.configure.signaling.lte.cell.cdrx.enable.set(cell_name = 'abc', enable =␣
↪False)
```

Enables or disables connected DRX.

param cell_name
No help available

param enable
No help available

6.3.4.7.2.10 Itimer

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:CDRX:ITIMer
```

class ItimerCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:CDRX:ITIMer
value: int = driver.configure.signaling.lte.cell.cdrx.itimer.get(cell_name =
↪'abc')
```

Configures the 'drx-InactivityTimer'.

param cell_name
No help available

return
timer: No help available

set(*cell_name: str*, *timer: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:CDRX:ITIMer
driver.configure.signaling.lte.cell.cdrx.itimer.set(cell_name = 'abc', timer =␣
↪1)
```

Configures the 'drx-InactivityTimer'.

param cell_name
No help available

param timer
No help available

6.3.4.7.2.11 Ldrx

class LdrxCls

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

Cloning the Group

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

Subgroups

6.3.4.7.2.12 Cycle

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:CDRX:LDRX:CYCLe
```

class CycleCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:CDRX:LDRX:CYCLe
value: int = driver.configure.signaling.lte.cell.cdrx.ldrx.cycle.get(cell_name␣
↪= 'abc')
```

Configures the duration of one long DRX cycle. The long DRX cycle duration must be divisible by the short DRX cycle duration.

param cell_name
No help available

return
cycle: No help available

set(*cell_name: str*, *cycle: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:CDRX:LDRX:CYCLe
driver.configure.signaling.lte.cell.cdrx.ldrx.cycle.set(cell_name = 'abc',␣
↪cycle = 1)
```

Configures the duration of one long DRX cycle. The long DRX cycle duration must be divisible by the short DRX cycle duration.

param cell_name
No help available

param cycle
No help available

6.3.4.7.2.13 Soffset

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:CDRX:LDRX:SOFFset
```

class SoffsetCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:CDRX:LDRX:SOFFset
value: int = driver.configure.signaling.lte.cell.cdrx.ldrx.soffset.get(cell_
↪name = 'abc')
```

Configures the drxStartOffset, shifting the DRX cycle start.

param cell_name
No help available

return
start_offset: No help available

set(*cell_name: str*, *start_offset: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:CDRX:LDRX:SOFFset
driver.configure.signaling.lte.cell.cdrx.ldrx.soffset.set(cell_name = 'abc',␣
↪start_offset = 1)
```

Configures the drxStartOffset, shifting the DRX cycle start.

param cell_name
No help available

param start_offset
No help available

6.3.4.7.2.14 OdTimer

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:CDRX:ODTimer
```

class OdTimerCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:CDRX:ODTimer
value: int = driver.configure.signaling.lte.cell.cdrx.odTimer.get(cell_name =
↪'abc')
```

Configures the 'onDurationTimer'.

param cell_name
No help available

return
timer: No help available

set(*cell_name: str*, *timer: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:CDRX:ODTimer
driver.configure.signaling.lte.cell.cdrx.odTimer.set(cell_name = 'abc', timer =␣
↪1)
```

Configures the 'onDurationTimer'.

param cell_name
No help available

param timer
No help available

6.3.4.7.2.15 Rtimer

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:CDRX:RTIMer
```

class RtimerCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:CDRX:RTIMer
value: int = driver.configure.signaling.lte.cell.cdrx.rtimer.get(cell_name =
↪'abc')
```

Configures the 'drx-RetransmissionTimer'.

param cell_name
No help available

return
timer: No help available

set(*cell_name: str*, *timer: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:CDRX:RTIMer
driver.configure.signaling.lte.cell.cdrx.rtimer.set(cell_name = 'abc', timer =␣
↪1)
```

Configures the 'drx-RetransmissionTimer'.

param cell_name
No help available

param timer
No help available

6.3.4.7.2.16 Sdrx

class SdrxCls

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

Cloning the Group

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

Subgroups

6.3.4.7.2.17 Cycle

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:CDRX:SDRX:CYCLe
```

class CycleCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:CDRX:SDRX:CYCLe
value: int = driver.configure.signaling.lte.cell.cdrx.sdrx.cycle.get(cell_name␣
↪= 'abc')
```

Configures the duration of one short DRX cycle. The long DRX cycle duration must be divisible by the short DRX cycle duration.

param cell_name
No help available

return
cycle: No help available

set(*cell_name: str*, *cycle: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:CDRX:SDRX:CYCLe
driver.configure.signaling.lte.cell.cdrx.sdrx.cycle.set(cell_name = 'abc',␣
↪cycle = 1)
```

Configures the duration of one short DRX cycle. The long DRX cycle duration must be divisible by the short DRX cycle duration.

param cell_name
No help available

param cycle
No help available

6.3.4.7.2.18 Enable

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:CDRX:SDRX:ENABle
```

class EnableCls

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

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:CDRX:SDRX:ENABle
value: bool = driver.configure.signaling.lte.cell.cdrx.sdrx.enable.get(cell_
↪name = 'abc')
```

Enables or disables short DRX cycles.

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:CDRX:SDRX:ENABle
driver.configure.signaling.lte.cell.cdrx.sdrx.enable.set(cell_name = 'abc',␣
↪enable = False)
```

Enables or disables short DRX cycles.

param cell_name
No help available

param enable
No help available

6.3.4.7.2.19 ScTimer

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:CDRX:SDRX:SCTimer
```

class ScTimerCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:CDRX:SDRX:SCTimer
value: int = driver.configure.signaling.lte.cell.cdrx.sdrx.scTimer.get(cell_
↪name = 'abc')
```

Configures the short cycle timer.

param cell_name
No help available

return
timer: No help available

set(*cell_name: str*, *timer: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:CDRX:SDRX:SCTimer
driver.configure.signaling.lte.cell.cdrx.sdrx.scTimer.set(cell_name = 'abc',␣
↪timer = 1)
```

Configures the short cycle timer.

param cell_name
No help available

param timer
No help available

6.3.4.7.2.20 Cmatrix

class CmatrixCls

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

Cloning the Group

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

Subgroups

6.3.4.7.2.21 Mode

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:CMATrix:MODE
```

class ModeCls

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

get(*cell_name: str*) → ModeFrecoveryB

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:CMATrix:MODE
value: enums.ModeFrecoveryB = driver.configure.signaling.lte.cell.cmatrix.mode.
↪get(cell_name = 'abc')
```

Applies the channel matrix selected via <Mode>.

param cell_name
No help available

return
mode: OFF: Without a fader in the signal path, IDENtity matrix. With a fader in the signal path, TGPP matrix. UDEFined: Matrix configured via other operating interface.

TGPP: Matrix as defined in 3GPP TS 36.101-4 annex B.1. HADamard: Hadamard matrix. IDENtity: Identity matrix.

set(*cell_name: str*, *mode: ModeFrecoveryB*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:CMATrix:MODE
driver.configure.signaling.lte.cell.cmatrix.mode.set(cell_name = 'abc', mode =␣
↪enums.ModeFrecoveryB.HADamard)
```

Applies the channel matrix selected via <Mode>.

param cell_name
No help available

param mode
OFF: Without a fader in the signal path, IDENtity matrix. With a fader in the signal path, TGPP matrix. UDEFined: Matrix configured via other operating interface. TGPP: Matrix as defined in 3GPP TS 36.101-4 annex B.1. HADamard: Hadamard matrix. IDENtity: Identity matrix.

6.3.4.7.2.22 CqiReporting

class CqiReportingCls

CqiReporting commands group definition. 6 total commands, 6 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.4.7.2.23 Cindex

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:CQIReporting:CINDex
```

class CindexCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:CQIReporting:CINDex
value: int = driver.configure.signaling.lte.cell.cqiReporting.cindex.get(cell_
↪name = 'abc')
```

Specifies the CQI/PMI configuration index (‘cqi-pmi-ConfigIndex’) .

param cell_name
No help available

return
index: No help available

set(*cell_name: str*, *index: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:CQIReporting:CINDex
driver.configure.signaling.lte.cell.cqiReporting.cindex.set(cell_name = 'abc',␣
↪index = 1)
```

Specifies the CQI/PMI configuration index (‘cqi-pmi-ConfigIndex’) .

param cell_name
No help available

param index
No help available

6.3.4.7.2.24 Findicator

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:CQIReporting:FINDicator
```

class FindicatorCls

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

get(*cell_name: str*) → FormatCqi

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:CQIReporting:FINDicator
value: enums.FormatCqi = driver.configure.signaling.lte.cell.cqiReporting.
↪findicator.get(cell_name = 'abc')
```

Configures the parameter ‘cqi-FormatIndicatorPeriodic’, signaled to the UE.

param cell_name
No help available

return
format_py: WB: wideband CQI reporting SB: subband CQI reporting

set(*cell_name: str*, *format_py: FormatCqi*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:CQIReporting:FINDicator
driver.configure.signaling.lte.cell.cqiReporting.findicator.set(cell_name = 'abc
↪', format_py = enums.FormatCqi.SB)
```

Configures the parameter ‘cqi-FormatIndicatorPeriodic’, signaled to the UE.

param cell_name
No help available

param format_py
WB: wideband CQI reporting SB: subband CQI reporting

6.3.4.7.2.25 PrEnable

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:CQIReporting:PRENable
```

class PrEnableCls

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

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:CQIReporting:PRENable
value: bool = driver.configure.signaling.lte.cell.cqiReporting.prEnable.
↪get(cell_name = 'abc')
```

Configures the reporting of PMI and RI values in addition to CQI values.

param cell_name
No help available

return
enable: ON: Request reporting of PMI/RI. OFF: Do not request reporting of PMI/RI.

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:CQIReporting:PRENable
driver.configure.signaling.lte.cell.cqiReporting.prEnable.set(cell_name = 'abc',
↪ enable = False)
```

Configures the reporting of PMI and RI values in addition to CQI values.

param cell_name
No help available

param enable
ON: Request reporting of PMI/RI. OFF: Do not request reporting of PMI/RI.

6.3.4.7.2.26 Rmode

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:CQIReporting:RMODe
```

class RmodeCls

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

get(*cell_name: str*) → ReportMode

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:CQIReporting:RMODe
value: enums.ReportMode = driver.configure.signaling.lte.cell.cqiReporting.
↪rmode.get(cell_name = 'abc')
```

Configures the parameter 'csi-ReportMode', signaled to the UE.

param cell_name
No help available

return
report_mode: S1: submode 1 S2: submode 2

set(*cell_name: str*, *report_mode: ReportMode*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:CQIReporting:RMODe
driver.configure.signaling.lte.cell.cqiReporting.rmode.set(cell_name = 'abc',␣
↪report_mode = enums.ReportMode.S1)
```

Configures the parameter 'csi-ReportMode', signaled to the UE.

param cell_name
No help available

param report_mode
S1: submode 1 S2: submode 2

6.3.4.7.2.27 Rtype

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:CQIReporting:RTYPe
```

class RtypeCls

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

get(*cell_name: str*) → ReportType

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:CQIReporting:RTYPe
value: enums.ReportType = driver.configure.signaling.lte.cell.cqiReporting.
↪rtype.get(cell_name = 'abc')
```

Selects the CQI reporting type.

param cell_name
No help available

return
type_py: OFF: no reporting PERiodic: periodic CQI reporting APERiodic: aperiodic CQI reporting

set(*cell_name: str*, *type_py: ReportType*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:CQIReporting:RTYPe
driver.configure.signaling.lte.cell.cqiReporting.rtype.set(cell_name = 'abc',␣
↪type_py = enums.ReportType.APERiodic)
```

Selects the CQI reporting type.

param cell_name
No help available

param type_py
OFF: no reporting PERiodic: periodic CQI reporting APERiodic: aperiodic CQI reporting

6.3.4.7.2.28 Sancqi

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:CQIReporting:SANCqi
```

class SancqiCls

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

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:CQIReporting:SANCqi
value: bool = driver.configure.signaling.lte.cell.cqiReporting.sancqi.get(cell_
↪name = 'abc')
```

Configures the parameter 'simultaneousAckNackAndCQI', signaled to the UE.

param cell_name
No help available

return
enable: ON: simultaneous transmission allowed OFF: not allowed

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:CQIReporting:SANCqi
driver.configure.signaling.lte.cell.cqiReporting.sancqi.set(cell_name = 'abc',␣
↪enable = False)
```

Configures the parameter 'simultaneousAckNackAndCQI', signaled to the UE.

param cell_name
No help available

param enable
ON: simultaneous transmission allowed OFF: not allowed

6.3.4.7.2.29 Harq

class HarqCls

Harq commands group definition. 7 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.4.7.2.30 Downlink

class DownlinkCls

Downlink commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.4.7.2.31 McsBehavior

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:HARQ:DL:MCSBehavior
```

class McsBehaviorCls

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

get(*cell_name: str*) → McsBehavior

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:HARQ:DL:MCSBehavior
value: enums.McsBehavior = driver.configure.signaling.lte.cell.harq.downlink.
↪mcsBehavior.get(cell_name = 'abc')
```

Defines the MCS selection for retransmissions.

param cell_name
No help available

return
behavior: Automatic, substitute with ReTx MCS, repeat initial MCS, replace if invalid transport block.

set(*cell_name: str*, *behavior: McsBehavior*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:HARQ:DL:MCSBehavior
driver.configure.signaling.lte.cell.harq.downlink.mcsBehavior.set(cell_name =
↪'abc', behavior = enums.McsBehavior.AUTO)
```

Defines the MCS selection for retransmissions.

param cell_name
No help available

param behavior
Automatic, substitute with ReTx MCS, repeat initial MCS, replace if invalid transport block.

6.3.4.7.2.32 PsOrder

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:HARQ:DL:PSORder
```

class PsOrderCls

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

get(*cell_name: str*) → PsOrder

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:HARQ:DL:PSORder
value: enums.PsOrder = driver.configure.signaling.lte.cell.harq.downlink.
↪psOrder.get(cell_name = 'abc')
```

Defines the scheduling order of the HARQ processes.

param cell_name
No help available

return
order: Round robin or subframe bound

set(*cell_name: str*, *order: PsOrder*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:HARQ:DL:PSORder
driver.configure.signaling.lte.cell.harq.downlink.psOrder.set(cell_name = 'abc',
↪ order = enums.PsOrder.RROBin)
```

Defines the scheduling order of the HARQ processes.

param cell_name
No help available

param order
Round robin or subframe bound

6.3.4.7.2.33 ReTx

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:HARQ:DL:RETX
```

class ReTxCls

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

class GetStruct

Response structure. Fields:

- Index: List[int]: Index of the entry to be configured (lowest index is 0) .
- Riv: List[enums.Riv]: RIV non-adaptive, new TX RIV
- Tb_1: List[int]: MCS value for first transport block
- Tb_2: List[int]: MCS value for second transport block

- Behavior: List[enums.ReTxBehavior]: Behavior for transport block size changes. Not applicable, flush HARQ buffer, retain HARQ buffer.

class SetStruct

Structure for setting input parameters. Fields:

- Cell_Name: str: No parameter help available
- Index: List[int]: Index of the entry to be configured (lowest index is 0) .
- Riv: List[enums.Riv]: RIV non-adaptive, new TX RIV
- Tb_1: List[int]: MCS value for first transport block
- Tb_2: List[int]: MCS value for second transport block
- Behavior: List[enums.ReTxBehavior]: Behavior for transport block size changes. Not applicable, flush HARQ buffer, retain HARQ buffer.

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:HARQ:DL:RETX
value: GetStruct = driver.configure.signaling.lte.cell.harq.downlink.reTx.
↪get(cell_name = 'abc')
```

Configures existing entries of the retransmission configuration.

param cell_name
No help available

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

set(*structure: SetStruct*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:HARQ:DL:RETX
structure = driver.configure.signaling.lte.cell.harq.downlink.reTx.SetStruct()
structure.Cell_Name: str = 'abc'
structure.Index: List[int] = [1, 2, 3]
structure.Riv: List[enums.Riv] = [Riv.NADaptive, Riv.NEW]
structure.Tb_1: List[int] = [1, 2, 3]
structure.Tb_2: List[int] = [1, 2, 3]
structure.Behavior: List[enums.ReTxBehavior] = [ReTxBehavior.FLUSh,␣
↪ReTxBehavior.RETain]
driver.configure.signaling.lte.cell.harq.downlink.reTx.set(structure)
```

Configures existing entries of the retransmission configuration.

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.lte.cell.harq.downlink.reTx.clone()
```

Subgroups

6.3.4.7.2.34 Behavior

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:HARQ:DL:RETX:BEHavior
```

class BehaviorCls

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

get(*cell_name: str*) → ReTxBehaviorB

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:HARQ:DL:RETX:BEHavior
value: enums.ReTxBehaviorB = driver.configure.signaling.lte.cell.harq.downlink.
↪reTx.behavior.get(cell_name = 'abc')
```

Defines a stop condition for retransmissions.

param cell_name
No help available

return
re_tx_behavior: - CONTinue: Send the maximum number of retransmissions. - STOP: Stop sending retransmissions when the UE answers with an ACK. - SNDMimo: Stop sending retransmissions when the UE answers with an ACK. For MIMO with two transport blocks, send no new data until HARQ for both transport blocks is complete (ACK or maximum retransmissions reached) . - SDTX: Stop sending retransmissions when DTX happens in the uplink.

set(*cell_name: str*, *re_tx_behavior: ReTxBehaviorB*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:HARQ:DL:RETX:BEHavior
driver.configure.signaling.lte.cell.harq.downlink.reTx.behavior.set(cell_name =
↪'abc', re_tx_behavior = enums.ReTxBehaviorB.CONTinue)
```

Defines a stop condition for retransmissions.

param cell_name
No help available

param re_tx_behavior

- CONTinue: Send the maximum number of retransmissions.
- STOP: Stop sending retransmissions when the UE answers with an ACK.
- SNDMimo: Stop sending retransmissions when the UE answers with an ACK. For MIMO with two transport blocks, send no new data until HARQ for both transport blocks is complete (ACK or maximum retransmissions reached) .
- SDTX: Stop sending retransmissions when DTX happens in the uplink.

6.3.4.7.2.35 Maximum

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:HARQ:DL:RETX:MAXimum
```

class MaximumCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:HARQ:DL:RETX:MAXimum
value: int = driver.configure.signaling.lte.cell.harq.downlink.reTx.maximum.
↪get(cell_name = 'abc')
```

Configures the maximum number of DL retransmissions.

param cell_name
No help available

return
max_re_tx: No help available

set(*cell_name: str*, *max_re_tx: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:HARQ:DL:RETX:MAXimum
driver.configure.signaling.lte.cell.harq.downlink.reTx.maximum.set(cell_name =
↪'abc', max_re_tx = 1)
```

Configures the maximum number of DL retransmissions.

param cell_name
No help available

param max_re_tx
No help available

6.3.4.7.2.36 RvSequence

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:HARQ:DL:RVSequence
```

class RvSequenceCls

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

class GetStruct

Response structure. Fields:

- Index: List[int]: Index of the entry to be configured (lowest index is 0) .
- Tb_1: List[enums.Version]: RV sequence for the first transport block, for QPSK and 16QAM.
- Tb_2: List[enums.Version]: RV sequence for the second transport block, for QPSK and 16QAM.
- Qam_64: List[enums.Version]: RV sequence for 64QAM and 256QAM, first and second transport block.

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:HARQ:DL:RVSequence
value: GetStruct = driver.configure.signaling.lte.cell.harq.downlink.rvSequence.
↪get(cell_name = 'abc')
```

Configures existing entries of user-defined RV sequences. If the mode is not user-defined, it is changed to user-defined. A query returns the sequences of the active mode, without changing the mode.

param cell_name
No help available

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

set(*cell_name: str*, *index: List[int]*, *tb_1: List[Version]*, *tb_2: List[Version]*, *qam_64: List[Version]*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:HARQ:DL:RVSequence
driver.configure.signaling.lte.cell.harq.downlink.rvSequence.set(cell_name =
↪'abc', index = [1, 2, 3], tb_1 = [Version.AUTO, Version.RV3], tb_2 = [Version.
↪AUTO, Version.RV3], qam_64 = [Version.AUTO, Version.RV3])
```

Configures existing entries of user-defined RV sequences. If the mode is not user-defined, it is changed to user-defined. A query returns the sequences of the active mode, without changing the mode.

param cell_name
No help available

param index
Index of the entry to be configured (lowest index is 0) .

param tb_1
RV sequence for the first transport block, for QPSK and 16QAM.

param tb_2
RV sequence for the second transport block, for QPSK and 16QAM.

param qam_64
RV sequence for 64QAM and 256QAM, first and second transport block.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.lte.cell.harq.downlink.rvSequence.clone()
```

Subgroups

6.3.4.7.2.37 Mode

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:HARQ:DL:RVSequence:MODE
```

class ModeCls

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

get(*cell_name: str*) → ModeRvs

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:HARQ:DL:RVSequence:MODE
value: enums.ModeRvs = driver.configure.signaling.lte.cell.harq.downlink.
↪rvSequence.mode.get(cell_name = 'abc')
```

Selects a mode for configuration of RV sequences.

param cell_name
No help available

return
mode: Auto, 3GPP 36.101, user-defined

set(*cell_name: str*, *mode: ModeRvs*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:HARQ:DL:RVSequence:MODE
driver.configure.signaling.lte.cell.harq.downlink.rvSequence.mode.set(cell_name␣
↪= 'abc', mode = enums.ModeRvs.AUTO)
```

Selects a mode for configuration of RV sequences.

param cell_name
No help available

param mode
Auto, 3GPP 36.101, user-defined

6.3.4.7.2.38 Info

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:INFO
```

class InfoCls

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

class GetStruct

Response structure. Fields:

- Phys_Cell_Id: int: No parameter help available
- Name_Ta: List[str]: No parameter help available

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:INFO
value: GetStruct = driver.configure.signaling.lte.cell.info.get(cell_name = 'abc
↪')
```

No command help available

param cell_name
No help available

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

6.3.4.7.2.39 Mconfig

class MconfigCls

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

Cloning the Group

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

Subgroups

6.3.4.7.2.40 Cdeployment

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:MCONfig:CDEPloyment
```

class CdeploymentCls

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

get(*cell_name: str*) → CellDeployment

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:MCONfig:CDEPloyment
value: enums.CellDeployment = driver.configure.signaling.lte.cell.mconfig.
↪cdeployment.get(cell_name = 'abc')
```

Selects whether the cell is a real cell or a virtual cell.

param cell_name
No help available

return
cell_deployment: No help available

set(*cell_name: str*, *cell_deployment: CellDeployment*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:MCONfig:CDEPloyment
driver.configure.signaling.lte.cell.mconfig.cdeployment.set(cell_name = 'abc',␣
↪cell_deployment = enums.CellDeployment.REAL)
```

Selects whether the cell is a real cell or a virtual cell.

param cell_name
No help available

param cell_deployment
No help available

6.3.4.7.2.41 CrSports

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:MCONfig:CRSPorts
```

class CrSportsCls

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

get(*cell_name: str*) → AntNoPorts

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:MCONfig:CRSPorts
value: enums.AntNoPorts = driver.configure.signaling.lte.cell.mconfig.crSports.
↪get(cell_name = 'abc')
```

Selects the maximum number of CRS antenna ports allowed in live mode.

param cell_name
No help available

return
ant_no_ports: No help available

set(*cell_name: str*, *ant_no_ports: AntNoPorts*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:MCONfig:CRSPorts
driver.configure.signaling.lte.cell.mconfig.crSports.set(cell_name = 'abc', ant_
↪no_ports = enums.AntNoPorts.P1)
```

Selects the maximum number of CRS antenna ports allowed in live mode.

param cell_name
No help available

param ant_no_ports
No help available

6.3.4.7.2.42 CsirsPorts

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:MCONfig:CSIRsports
```

class CsirsPortsCls

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

get(*cell_name: str*) → AntNoPortsB

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:MCONfig:CSIRsports
value: enums.AntNoPortsB = driver.configure.signaling.lte.cell.mconfig.
↪csirsPorts.get(cell_name = 'abc')
```

Selects the maximum number of CSI-RS antenna ports allowed in live mode.

param cell_name
No help available

return
ant_no_ports: No help available

set(*cell_name: str*, *ant_no_ports: AntNoPortsB*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:MCONfig:CSIRsports
driver.configure.signaling.lte.cell.mconfig.csirsPorts.set(cell_name = 'abc',␣
↪ant_no_ports = enums.AntNoPortsB.P1)
```

Selects the maximum number of CSI-RS antenna ports allowed in live mode.

param cell_name
No help available

param ant_no_ports
No help available

6.3.4.7.2.43 DlOnly

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:MCONfig:DLONly
```

class DlOnlyCls

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

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:MCONfig:DLONly
value: bool = driver.configure.signaling.lte.cell.mconfig.dlOnly.get(cell_name␣
↪= 'abc')
```

Selects whether UL is forbidden for a cell in live mode.

param cell_name
No help available

return
enable: ON: only DL OFF: UL and DL allowed

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:MCONfig:DLONly
driver.configure.signaling.lte.cell.mconfig.dlOnly.set(cell_name = 'abc',␣
↪enable = False)
```

Selects whether UL is forbidden for a cell in live mode.

param cell_name
No help available

param enable
ON: only DL OFF: UL and DL allowed

6.3.4.7.2.44 Modulation

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:MCONfig:MODulation
```

class ModulationCls

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

get(*cell_name: str*) → Modulation

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:MCONfig:MODulation
value: enums.Modulation = driver.configure.signaling.lte.cell.mconfig.
↪modulation.get(cell_name = 'abc')
```

Selects the maximum UL modulation scheme allowed in live mode.

param cell_name
No help available

return
modulation: No help available

set(*cell_name: str*, *modulation: Modulation*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:MCONfig:MODulation
driver.configure.signaling.lte.cell.mconfig.modulation.set(cell_name = 'abc',␣
↪modulation = enums.Modulation.BPSK)
```

Selects the maximum UL modulation scheme allowed in live mode.

param cell_name
No help available

param modulation
No help available

6.3.4.7.2.45 Mimo

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:MIMO
```

class MimoCls

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

get(*cell_name: str*) → LteMimoScheme

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:MIMO
value: enums.LteMimoScheme = driver.configure.signaling.lte.cell.mimo.get(cell_
↪name = 'abc')
```

No command help available

param cell_name
No help available

return
lte_mimo_scheme: No help available

set(*cell_name: str*, *lte_mimo_scheme: LteMimoScheme*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:MIMO
driver.configure.signaling.lte.cell.mimo.set(cell_name = 'abc', lte_mimo_scheme␣
↪= enums.LteMimoScheme.M2N)
```

No command help available

param cell_name
No help available

param lte_mimo_scheme
No help available

6.3.4.7.2.46 Pcid

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:PCID
```

class PcidCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:PCID
value: int = driver.configure.signaling.lte.cell.pcid.get(cell_name = 'abc')
```

Defines the physical cell ID.

param cell_name
No help available

return
idn: No help available

set(*cell_name: str*, *idn: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:PCID
driver.configure.signaling.lte.cell.pcid.set(cell_name = 'abc', idn = 1)
```

Defines the physical cell ID.

param cell_name
No help available

param idn
No help available

6.3.4.7.2.47 Pcycle

class PcycleCls

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

Cloning the Group

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

Subgroups

6.3.4.7.2.48 Pcycle

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:PCYCle:PCYCle
```

class PcycleCls

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

get(*cell_name: str*) → PagingCycle

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:PCYCle:PCYCle
value: enums.PagingCycle = driver.configure.signaling.lte.cell.pcycle.pcycle.
↪get(cell_name = 'abc')
```

Selects the paging cycle in radio frames.

param cell_name
No help available

return
paging_cycle: No help available

set(*cell_name: str*, *paging_cycle: PagingCycle*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:PCYCle:PCYCle
driver.configure.signaling.lte.cell.pcycle.pcycle.set(cell_name = 'abc', paging_
↪cycle = enums.PagingCycle.P128)
```

Selects the paging cycle in radio frames.

param cell_name
No help available

param paging_cycle
No help available

6.3.4.7.2.49 PfOffset

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:PCYCle:PFOFfset
```

class PfOffsetCls

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

get(*cell_name: str*) → FramesOffset

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:PCYCle:PFOFfset
value: enums.FramesOffset = driver.configure.signaling.lte.cell.pcycle.pfOffset.
↪get(cell_name = 'abc')
```

Configures the field 'nB', used by the UE as input for the calculation of paging radio frame and paging occasion.

param cell_name
No help available

return
frames_offset: T4T, T2T, T1T: 4, 2, 1 T2 | T4 | T8 | T16 | T32: 1/2, 1/4, 1/8, 1/16, 1/32

set(*cell_name: str*, *frames_offset: FramesOffset*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:PCYCle:PFOFfset
driver.configure.signaling.lte.cell.pcycle.pfOffset.set(cell_name = 'abc',␣
↪frames_offset = enums.FramesOffset.T16)
```

Configures the field 'nB', used by the UE as input for the calculation of paging radio frame and paging occasion.

param cell_name
No help available

param frames_offset
T4T, T2T, T1T: 4, 2, 1 T2 | T4 | T8 | T16 | T32: 1/2, 1/4, 1/8, 1/16, 1/32

6.3.4.7.2.50 Power

class PowerCls

Power commands group definition. 40 total commands, 3 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.4.7.2.51 Control

class ControlCls

Control commands group definition. 7 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.4.7.2.52 Channel

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:POWer:CONTrol:CHANnel
```

class ChannelCls

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

get(*cell_name: str*) → SrcType

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:CONTrol:CHANnel
value: enums.SrcType = driver.configure.signaling.lte.cell.power.control.
↪channel.get(cell_name = 'abc')
```

Selects the uplink channel types to which the power control commands are applied.

param cell_name
No help available

return
type_py: PUSC: PUSCH PUCC: PUCCH PUPU: PUSCH and PUCCH

set(*cell_name: str*, *type_py: SrcType*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:CONTrol:CHANnel
driver.configure.signaling.lte.cell.power.control.channel.set(cell_name = 'abc',
↪ type_py = enums.SrcType.PUCC)
```

Selects the uplink channel types to which the power control commands are applied.

param cell_name
No help available

param type_py
PUSC: PUSCH PUCC: PUCCH PUPU: PUSCH and PUCCH

6.3.4.7.2.53 TpControl

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:POWer:CONTrol:TPControl
```

class TpControlCls

TpControl commands group definition. 6 total commands, 2 Subgroups, 1 group commands

get(*cell_name: str*) → Control

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:CONTrol:TPControl
value: enums.Control = driver.configure.signaling.lte.cell.power.control.
↪tpControl.get(cell_name = 'abc')
```

Selects the pattern of TPC commands to be sent to the UE.

param cell_name
No help available

return
control: Keep, min, max, closed loop, TPC pattern.

set(*cell_name: str*, *control: Control*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:CONTrol:TPControl
driver.configure.signaling.lte.cell.power.control.tpControl.set(cell_name = 'abc
↪', control = enums.Control.CLOop)
```

Selects the pattern of TPC commands to be sent to the UE.

param cell_name
No help available

param control
Keep, min, max, closed loop, TPC pattern.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.lte.cell.power.control.tpControl.clone()
```

Subgroups

6.3.4.7.2.54 Cloop

class CloopCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.lte.cell.power.control.tpControl.cloop.clone()
```

Subgroups

6.3.4.7.2.55 Tolerance

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:POWer:CONTrol:TPControl:CLOop:TOLerance
```

class ToleranceCls

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

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:CONTrol:TPControl:CLOop:TOLerance
value: float = driver.configure.signaling.lte.cell.power.control.tpControl.
↪cloop.tolerance.get(cell_name = 'abc')
```

Defines the tolerance for closed-loop power control.

param cell_name
No help available

return
tolerance: No help available

set(*cell_name: str*, *tolerance: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:CONTrol:TPControl:CLOop:TOLerance
driver.configure.signaling.lte.cell.power.control.tpControl.cloop.tolerance.
↪set(cell_name = 'abc', tolerance = 1.0)
```

Defines the tolerance for closed-loop power control.

param cell_name
No help available

param tolerance
No help available

6.3.4.7.2.56 Tpower

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:POWer:CONTrol:TPControl:CLOop:TPOWer
```

class TpowerCls

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

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:CONTrol:TPControl:CLOop:TPOWer
value: float = driver.configure.signaling.lte.cell.power.control.tpControl.
↪cloop.tpower.get(cell_name = 'abc')
```

Defines the target power for closed-loop power control.

param cell_name
No help available

return
power: No help available

set(*cell_name: str*, *power: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:CONTrol:TPControl:CLOop:TPOWer
driver.configure.signaling.lte.cell.power.control.tpControl.cloop.tpower.
↪set(cell_name = 'abc', power = 1.0)
```

Defines the target power for closed-loop power control.

param cell_name
No help available

param power
No help available

6.3.4.7.2.57 Pattern

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:POWer:CONTrol:TPControl:PATTern
```

class PatternCls

Pattern commands group definition. 3 total commands, 1 Subgroups, 1 group commands

get(*cell_name: str*) → TypeB

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:CONTrol:TPControl:PATTern
value: enums.TypeB = driver.configure.signaling.lte.cell.power.control.
↪tpControl.pattern.get(cell_name = 'abc')
```

No command help available

param cell_name
No help available

return
type_py: No help available

set(*cell_name: str*, *type_py: TypeB*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:CONTrol:TPControl:PATTern
driver.configure.signaling.lte.cell.power.control.tpControl.pattern.set(cell_
↪name = 'abc', type_py = enums.TypeB.UDEFined)
```

No command help available

param cell_name
No help available

param type_py
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.lte.cell.power.control.tpControl.pattern.clone()
```

Subgroups

6.3.4.7.2.58 UserDefined

class UserDefinedCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.lte.cell.power.control.tpControl.pattern.userDefined.
↪clone()
```

Subgroups

6.3.4.7.2.59 Mode

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:POWer:CONTrol:TPControl:PATTern:UDEFined:MODE
```

class ModeCls

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

get(*cell_name: str*) → Repeat

```
# SCPI:␣
↪[CONFigure]:SIGNaling:LTE:CELL:POWer:CONTrol:TPControl:PATTern:UDEFined:MODE
value: enums.Repeat = driver.configure.signaling.lte.cell.power.control.
↪tpControl.pattern.userDefined.mode.get(cell_name = 'abc')
```

Selects the mode for execution of a user-defined TPC pattern.

param cell_name
No help available

return
mode: No help available

set(*cell_name: str*, *mode: Repeat*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:LTE:CELL:POWer:CONTrol:TPControl:PATTern:UDEFined:MODE
driver.configure.signaling.lte.cell.power.control.tpControl.pattern.userDefined.
↪mode.set(cell_name = 'abc', mode = enums.Repeat.CONTinuous)
```

Selects the mode for execution of a user-defined TPC pattern.

param cell_name
No help available

param mode
No help available

6.3.4.7.2.60 Pattern

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:POWer:CONTrol:TPControl:PATTern:UDEFined:PATTern
```

class PatternCls

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

get(*cell_name: str*) → List[Pattern]

```
# SCPI:␣
↪[CONFigure]:SIGNaling:LTE:CELL:POWer:CONTrol:TPControl:PATTern:UDEFined:PATTern
value: List[enums.Pattern] = driver.configure.signaling.lte.cell.power.control.
↪tpControl.pattern.userDefined.pattern.get(cell_name = 'abc')
```

Configures a user-defined TPC pattern as a sequence of commands.

param cell_name
No help available

return
pattern: Comma-separated list of commands D1: -1 dB KEEP: 0 dB U1: +1 dB U3: +3 dB A single NAV is returned if no pattern is defined.

set(*cell_name: str*, *pattern: List[Pattern]*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:LTE:CELL:POWer:CONTrol:TPControl:PATTern:UDEFined:PATTern
driver.configure.signaling.lte.cell.power.control.tpControl.pattern.userDefined.
↪pattern.set(cell_name = 'abc', pattern = [Pattern.D1, Pattern.U3])
```

Configures a user-defined TPC pattern as a sequence of commands.

param cell_name
No help available

param pattern
Comma-separated list of commands D1: -1 dB KEEP: 0 dB U1: +1 dB U3: +3 dB A single NAV is returned if no pattern is defined.

6.3.4.7.2.61 Downlink

class DownlinkCls

Downlink commands group definition. 13 total commands, 5 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.4.7.2.62 Maximum

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:POWer:DL:MAXimum
```

class MaximumCls

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

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:DL:MAXimum
value: float = driver.configure.signaling.lte.cell.power.downlink.maximum.
↪get(cell_name = 'abc')
```

Defines the maximum cell power.

param cell_name
No help available

return
max_cell_power: No help available

set(*cell_name: str*, *max_cell_power: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:DL:MAXimum
driver.configure.signaling.lte.cell.power.downlink.maximum.set(cell_name = 'abc
↪', max_cell_power = 1.0)
```

Defines the maximum cell power.

param cell_name
No help available

param max_cell_power
No help available

6.3.4.7.2.63 Ocng

class OcngCls

Ocng commands group definition. 4 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.lte.cell.power.downlink.ocng.clone()
```

Subgroups

6.3.4.7.2.64 Enable

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:POWer:DL:OCNG:ENABle
```

class EnableCls

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

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:DL:OCNG:ENABle
value: bool = driver.configure.signaling.lte.cell.power.downlink.ocng.enable.
↪get(cell_name = 'abc')
```

Enables or disables the OFDMA channel noise generator (OCNG) .

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:DL:OCNG:ENABle
driver.configure.signaling.lte.cell.power.downlink.ocng.enable.set(cell_name =
↪'abc', enable = False)
```

Enables or disables the OFDMA channel noise generator (OCNG) .

param cell_name
No help available

param enable
No help available

6.3.4.7.2.65 Pdcch

class PdcchCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.lte.cell.power.downlink.ocng.pdcch.clone()
```

Subgroups

6.3.4.7.2.66 Poffset

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:POWer:DL:OCNG:PDCCh:POFFset
```

class PoffsetCls

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

class GetStruct

Response structure. Fields:

- Mode: enums.ModeD: No parameter help available
- Value: float: Power level relative to the RS EPRE.

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:DL:OCNG:PDCCh:POFFset
value: GetStruct = driver.configure.signaling.lte.cell.power.downlink.ocng.
↪pdcch.poffset.get(cell_name = 'abc')
```

Defines the power level of the PDCCH for OCNG.

param cell_name
No help available

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

set(*cell_name: str*, *mode: ModeD*, *value: float = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:DL:OCNG:PDCCh:POFFset
driver.configure.signaling.lte.cell.power.downlink.ocng.pdcch.poffset.set(cell_
↪name = 'abc', mode = enums.ModeD.MAX, value = 1.0)
```

Defines the power level of the PDCCH for OCNG.

param cell_name
No help available

param mode
No help available

param value
Power level relative to the RS EPRE.

6.3.4.7.2.67 Pdsch

class PdschCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.lte.cell.power.downlink.ocng.pdsch.clone()
```

Subgroups

6.3.4.7.2.68 Modulation

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:POWer:DL:OCNG:PDSCh:MODulation
```

class ModulationCls

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

get(*cell_name: str*) → Modulation

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:DL:OCNG:PDSCh:MODulation
value: enums.Modulation = driver.configure.signaling.lte.cell.power.downlink.
↪ocng.pdsch.modulation.get(cell_name = 'abc')
```

Selects the modulation scheme of the PDSCH for OCNG.

param cell_name
No help available

return
modulation: No help available

set(*cell_name: str*, *modulation: Modulation*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:DL:OCNG:PDSCh:MODulation
driver.configure.signaling.lte.cell.power.downlink.ocng.pdsch.modulation.
↪set(cell_name = 'abc', modulation = enums.Modulation.BPSK)
```

Selects the modulation scheme of the PDSCH for OCNG.

param cell_name
No help available

param modulation
No help available

6.3.4.7.2.69 Poffset

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:POWer:DL:OCNG:PDSCh:POFFset
```

class PoffsetCls

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

class GetStruct

Response structure. Fields:

- Mode: enums.ModeD: No parameter help available
- Value: float: Power level relative to the RS EPRE

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:DL:OCNG:PDSCh:POFFset
value: GetStruct = driver.configure.signaling.lte.cell.power.downlink.ocng.
↪pdsch.poffset.get(cell_name = 'abc')
```

Defines the power level of the PDSCH for OCNG.

param cell_name
No help available

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

set(*cell_name: str*, *mode: ModeD*, *value: float = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:DL:OCNG:PDSCh:POFFset
driver.configure.signaling.lte.cell.power.downlink.ocng.pdsch.poffset.set(cell_
↪name = 'abc', mode = enums.ModeD.MAX, value = 1.0)
```

Defines the power level of the PDSCH for OCNG.

param cell_name
No help available

param mode
No help available

param value
Power level relative to the RS EPRE

6.3.4.7.2.70 Offset

class OffsetCls

Offset commands group definition. 6 total commands, 6 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.lte.cell.power.downlink.offset.clone()
```

Subgroups

6.3.4.7.2.71 Pbch

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:POWer:DL:OFFSet:PBCH
```

class PbchCls

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

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:DL:OFFSet:PBCH
value: float = driver.configure.signaling.lte.cell.power.downlink.offset.pbch.
↪get(cell_name = 'abc')
```

Power level of the PBCH relative to the RS EPRE setting.

param cell_name
No help available

return
decibel: No help available

set(*cell_name: str*, *decibel: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:DL:OFFSet:PBCH
driver.configure.signaling.lte.cell.power.downlink.offset.pbch.set(cell_name =
↪'abc', decibel = 1.0)
```

Power level of the PBCH relative to the RS EPRE setting.

param cell_name
No help available

param decibel
No help available

6.3.4.7.2.72 Pcfich

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:POWer:DL:OFFSet:PCFich
```

class PcfichCls

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

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:DL:OFFSet:PCFich
value: float = driver.configure.signaling.lte.cell.power.downlink.offset.pcfich.
↪get(cell_name = 'abc')
```

Power level of the PCFICH relative to the RS EPRE setting.

param cell_name
No help available

return
decibel: No help available

set(*cell_name: str*, *decibel: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:DL:OFFSet:PCFich
driver.configure.signaling.lte.cell.power.downlink.offset.pcfich.set(cell_name␣
↪= 'abc', decibel = 1.0)
```

Power level of the PCFICH relative to the RS EPRE setting.

param cell_name
No help available

param decibel
No help available

6.3.4.7.2.73 Pdcch

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:POWer:DL:OFFSet:PDCCh
```

class PdcchCls

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

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:DL:OFFSet:PDCCh
value: float = driver.configure.signaling.lte.cell.power.downlink.offset.pdcch.
↪get(cell_name = 'abc')
```

Power level of the PDCCH relative to the RS EPRE setting.

param cell_name
No help available

return
decibel: No help available

set(*cell_name: str*, *decibel: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:DL:OFFSet:PDCCh
driver.configure.signaling.lte.cell.power.downlink.offset.pdcch.set(cell_name =
↪'abc', decibel = 1.0)
```

Power level of the PDCCH relative to the RS EPRE setting.

param cell_name
No help available

param decibel
No help available

6.3.4.7.2.74 Pss

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:POWer:DL:OFFSet:PSS
```

class PssCls

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

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:DL:OFFSet:PSS
value: float = driver.configure.signaling.lte.cell.power.downlink.offset.pss.
↪get(cell_name = 'abc')
```

Power level of the PSS relative to the RS EPRE setting.

param cell_name
No help available

return
decibel: No help available

set(*cell_name: str*, *decibel: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:DL:OFFSet:PSS
driver.configure.signaling.lte.cell.power.downlink.offset.pss.set(cell_name =
↪'abc', decibel = 1.0)
```

Power level of the PSS relative to the RS EPRE setting.

param cell_name
No help available

param decibel
No help available

6.3.4.7.2.75 Rs

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:POWer:DL:OFFSet:RS
```

class RsCls

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

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:DL:OFFSet:RS
value: float = driver.configure.signaling.lte.cell.power.downlink.offset.rs.
↪get(cell_name = 'abc')
```

Power level of the RS relative to the RS EPRE setting.

param cell_name
No help available

return
decibel: No help available

set(*cell_name: str*, *decibel: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:DL:OFFSet:RS
driver.configure.signaling.lte.cell.power.downlink.offset.rs.set(cell_name =
↪'abc', decibel = 1.0)
```

Power level of the RS relative to the RS EPRE setting.

param cell_name
No help available

param decibel
No help available

6.3.4.7.2.76 Sss

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:POWer:DL:OFFSet:SSS
```

class SssCls

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

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:DL:OFFSet:SSS
value: float = driver.configure.signaling.lte.cell.power.downlink.offset.sss.
↪get(cell_name = 'abc')
```

Power level of the SSS relative to the RS EPRE setting.

param cell_name
No help available

return
decibel: No help available

set(*cell_name: str*, *decibel: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:DL:OFFSet:SSS
driver.configure.signaling.lte.cell.power.downlink.offset.sss.set(cell_name =
↪'abc', decibel = 1.0)
```

Power level of the SSS relative to the RS EPRE setting.

param cell_name
No help available

param decibel
No help available

6.3.4.7.2.77 Reference

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:POWer:DL:REFerence
```

class ReferenceCls

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

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:DL:REFerence
value: float = driver.configure.signaling.lte.cell.power.downlink.reference.
↪get(cell_name = 'abc')
```

Configures the reference signal power.

param cell_name
No help available

return
ref_signal_power: No help available

set(*cell_name: str*, *ref_signal_power: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:DL:REFerence
driver.configure.signaling.lte.cell.power.downlink.reference.set(cell_name =
↪'abc', ref_signal_power = 1.0)
```

Configures the reference signal power.

param cell_name
No help available

param ref_signal_power
No help available

6.3.4.7.2.78 Rsepre

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:POWer:DL:RSEPre
```

class RsepreCls

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

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:DL:RSEPre
value: float = driver.configure.signaling.lte.cell.power.downlink.rsepre.
↪get(cell_name = 'abc')
```

Defines the energy per resource element (EPRE) of the cell-specific reference signal (C-RS) .

param cell_name
No help available

return
rs_erpe: No help available

set(*cell_name: str*, *rs_erpe: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:DL:RSEPre
driver.configure.signaling.lte.cell.power.downlink.rsepre.set(cell_name = 'abc',
↪ rs_erpe = 1.0)
```

Defines the energy per resource element (EPRE) of the cell-specific reference signal (C-RS) .

param cell_name
No help available

param rs_erpe
No help available

6.3.4.7.2.79 Uplink

class UplinkCls

Uplink commands group definition. 20 total commands, 17 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.4.7.2.80 Alpha

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:POWer:UL:ALPHa
```

class AlphaCls

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

get(*cell_name: str*) → Alpha

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:UL:ALPHa
value: enums.Alpha = driver.configure.signaling.lte.cell.power.uplink.alpha.
↪get(cell_name = 'abc')
```

Sets the UL power control parameter alpha.

param cell_name
No help available

return
alpha: Axy means x.y.

set(*cell_name: str*, *alpha: Alpha*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:UL:ALPHa
driver.configure.signaling.lte.cell.power.uplink.alpha.set(cell_name = 'abc',␣
↪alpha = enums.Alpha.A00)
```

Sets the UL power control parameter alpha.

param cell_name
No help available

param alpha
Axy means x.y.

6.3.4.7.2.81 Auto

class AutoCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.lte.cell.power.uplink.auto.clone()
```

Subgroups

6.3.4.7.2.82 RlOffset

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:POWer:UL:AUTO:RLOFfset
```

class RlOffsetCls

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

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:UL:AUTO:RLOFfset
value: float = driver.configure.signaling.lte.cell.power.uplink.auto.rlOffset.
↪get(cell_name = 'abc')
```

Sets the reference level offset for automatic configuration of expected UL power.

param cell_name
No help available

return
ref_level_offset: No help available

set(*cell_name: str*, *ref_level_offset: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:UL:AUTO:RLOFfset
driver.configure.signaling.lte.cell.power.uplink.auto.rlOffset.set(cell_name =
↪'abc', ref_level_offset = 1.0)
```

Sets the reference level offset for automatic configuration of expected UL power.

param cell_name
No help available

param ref_level_offset
No help available

6.3.4.7.2.83 Rsource

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:POWer:UL:AUTO:RSOurce
```

class RsourceCls

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

get(*cell_name: str*) → SrcType

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:UL:AUTO:RSOurce
value: enums.SrcType = driver.configure.signaling.lte.cell.power.uplink.auto.
↪rsource.get(cell_name = 'abc')
```

Sets the reference source for automatic configuration of expected UL power.

param cell_name
No help available

return
ref_source: PUSC: PUSCH PUCC: PUCCH PUPU: PUCCH and PUSCH

set(*cell_name: str*, *ref_source: SrcType*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:UL:AUTO:RSOurce
driver.configure.signaling.lte.cell.power.uplink.auto.rsource.set(cell_name =
↪'abc', ref_source = enums.SrcType.PUCC)
```

Sets the reference source for automatic configuration of expected UL power.

param cell_name
No help available

param ref_source
PUSC: PUSCH PUCC: PUCCH PUPU: PUCCH and PUSCH

6.3.4.7.2.84 Cindex

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:POWer:UL:CINDex
```

class CindexCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:UL:CINDex
value: int = driver.configure.signaling.lte.cell.power.uplink.cindex.get(cell_
↪name = 'abc')
```

Sets the PRACH configuration index to be used by the UE.

param cell_name
No help available

return
index: No help available

set(*cell_name: str*, *index: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:UL:CINDex
driver.configure.signaling.lte.cell.power.uplink.cindex.set(cell_name = 'abc',␣
↪index = 1)
```

Sets the PRACH configuration index to be used by the UE.

param cell_name
No help available

param index
No help available

6.3.4.7.2.85 Cmode

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:POWer:UL:CMODe
```

class CmodeCls

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

get(*cell_name: str*) → ConfigMode

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:UL:CMODe
value: enums.ConfigMode = driver.configure.signaling.lte.cell.power.uplink.
↪cmode.get(cell_name = 'abc')
```

Selects a configuration mode for the expected UL level.

param cell_name
No help available

return
config_mode: Automatic configuration or user-defined configuration

set(*cell_name: str*, *config_mode: ConfigMode*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:UL:CMODe
driver.configure.signaling.lte.cell.power.uplink.cmode.set(cell_name = 'abc',␣
↪config_mode = enums.ConfigMode.AUTO)
```

Selects a configuration mode for the expected UL level.

param cell_name
No help available

param config_mode
Automatic configuration or user-defined configuration

6.3.4.7.2.86 Epre

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:POWer:UL:EPRE
```

class EpreCls

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

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:UL:EPRE
value: float = driver.configure.signaling.lte.cell.power.uplink.epre.get(cell_
↪name = 'abc')
```

Sets the maximum EPRE expected in the UL, for user-defined configuration. For automatic configuration, you can query the value.

param cell_name
No help available

return
max_exp_erpe: No help available

set(*cell_name: str*, *max_exp_erpe: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:UL:EPRE
driver.configure.signaling.lte.cell.power.uplink.epre.set(cell_name = 'abc',␣
↪max_exp_erpe = 1.0)
```

Sets the maximum EPRE expected in the UL, for user-defined configuration. For automatic configuration, you can query the value.

param cell_name
No help available

param max_exp_erpe
No help available

6.3.4.7.2.87 Fcoefficient

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:POWer:UL:FCOefficient
```

class FcoefficientCls

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

get(*cell_name: str*) → FilterCoeff

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:UL:FCOefficient
value: enums.FilterCoeff = driver.configure.signaling.lte.cell.power.uplink.
↪fcoefficient.get(cell_name = 'abc')
```

Sets the parameter 'filterCoefficient', signaled to the UE as an uplink power control parameter.

param cell_name
No help available

return
filter_coeff: No help available

set(*cell_name: str*, *filter_coeff: FilterCoeff*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:UL:FCOefficient
driver.configure.signaling.lte.cell.power.uplink.fcoefficient.set(cell_name =
↪'abc', filter_coeff = enums.FilterCoeff.FC0)
```

Sets the parameter 'filterCoefficient', signaled to the UE as an uplink power control parameter.

param cell_name
No help available

param filter_coeff
No help available

6.3.4.7.2.88 HsFlag

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:POWer:UL:HSFLag
```

class HsFlagCls

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

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:UL:HSFLag
value: bool = driver.configure.signaling.lte.cell.power.uplink.hsFlag.get(cell_
↪name = 'abc')
```

Sets the parameter 'highSpeedFlag', signaled to the UE

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:UL:HSFLag
driver.configure.signaling.lte.cell.power.uplink.hsFlag.set(cell_name = 'abc',␣
↪enable = False)
```

Sets the parameter 'highSpeedFlag', signaled to the UE

param cell_name
No help available

param enable
No help available

6.3.4.7.2.89 IpPreambles

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:POWer:UL:IPPReambles
```

class IpPreamblesCls

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

class GetStruct

Response structure. Fields:

- Ignore_Prach_Mode: enums.IgnorePrachMode: IALLways: ignore all preambles IXTimes: ignore NoIgnored preambles RALLways: respond to all preambles
- No_Ignored: int: Number of preambles to be ignored for IgnorePrachMode = IXTimes.

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:UL:IPPReambles
value: GetStruct = driver.configure.signaling.lte.cell.power.uplink.ipPreambles.
↪get(cell_name = 'abc')
```

Selects the behavior of the signaling application when receiving preambles from the UE. The setting is synchronized over all LTE cells (identical values for all LTE cells) .

param cell_name
No help available

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

set(*cell_name: str*, *ignore_prach_mode: IgnorePrachMode*, *no_ignored: int = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:UL:IPPReambles
driver.configure.signaling.lte.cell.power.uplink.ipPreambles.set(cell_name =
↪'abc', ignore_prach_mode = enums.IgnorePrachMode.IALLways, no_ignored = 1)
```

Selects the behavior of the signaling application when receiving preambles from the UE. The setting is synchronized over all LTE cells (identical values for all LTE cells) .

param cell_name
No help available

param ignore_prach_mode
IALLways: ignore all preambles IXTimes: ignore NoIgnored preambles RALLways: respond to all preambles

param no_ignored
Number of preambles to be ignored for IgnorePrachMode = IXTimes.

6.3.4.7.2.90 IptPower

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:POWer:UL:IPTPower
```

class IptPowerCls

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

get(*cell_name: str*) → Power

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:UL:IPTPower
value: enums.Power = driver.configure.signaling.lte.cell.power.uplink.iptPower.
↪get(cell_name = 'abc')
```

Sets the parameter 'preambleInitialReceivedTargetPower', signaled to the UE as a common RACH parameter.

param cell_name
No help available

return
power: Negative dBm value (-90 dBm to -120 dBm)

set(*cell_name: str*, *power: Power*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:UL:IPTPower
driver.configure.signaling.lte.cell.power.uplink.iptPower.set(cell_name = 'abc',
↪ power = enums.Power.P100)
```

Sets the parameter 'preambleInitialReceivedTargetPower', signaled to the UE as a common RACH parameter.

param cell_name
No help available

param power
Negative dBm value (-90 dBm to -120 dBm)

6.3.4.7.2.91 LrsIndex

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:POWer:UL:LRSindex
```

class LrsIndexCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:UL:LRSindex
value: int = driver.configure.signaling.lte.cell.power.uplink.lrsIndex.get(cell_
↪name = 'abc')
```

Sets the parameter 'rootSequenceIndex', signaled to the UE.

param cell_name
No help available

return
index: No help available

set(*cell_name: str*, *index: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:UL:LRSindex
driver.configure.signaling.lte.cell.power.uplink.lrsIndex.set(cell_name = 'abc',
↪ index = 1)
```

Sets the parameter 'rootSequenceIndex', signaled to the UE.

param cell_name
No help available

param index
No help available

6.3.4.7.2.92 Pmax

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:POWer:UL:PMAX
```

class PmaxCls

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

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:UL:PMAX
value: float or bool = driver.configure.signaling.lte.cell.power.uplink.pmax.
↪get(cell_name = 'abc')
```

Sets the UL power control parameter 'p-Max'.

param cell_name
No help available

return
power: (float or boolean) OFF means that the parameter is not signaled.

set(*cell_name: str*, *power: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:UL:PMAX
driver.configure.signaling.lte.cell.power.uplink.pmax.set(cell_name = 'abc',␣
↪power = 1.0)
```

Sets the UL power control parameter 'p-Max'.

param cell_name
No help available

param power
(float or boolean) OFF means that the parameter is not signaled.

6.3.4.7.2.93 PrStep

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:POWer:UL:PRSTep
```

class PrStepCls

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

get(*cell_name: str*) → PwrRampingStepA

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:UL:PRSTep
value: enums.PwrRampingStepA = driver.configure.signaling.lte.cell.power.uplink.
↪prStep.get(cell_name = 'abc')
```

Defines the transmit power difference between two consecutive preambles (power ramping) .

param cell_name
No help available

return
pwr_ramping_step: Step size in dB (0 dB to 6 dB)

set(*cell_name: str*, *pwr_ramping_step: PwrRampingStepA*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:UL:PRSTep
driver.configure.signaling.lte.cell.power.uplink.prStep.set(cell_name = 'abc',␣
↪pwr_ramping_step = enums.PwrRampingStepA.S0)
```

Defines the transmit power difference between two consecutive preambles (power ramping) .

param cell_name
No help available

param pwr_ramping_step
Step size in dB (0 dB to 6 dB)

6.3.4.7.2.94 PsRsOffset

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:POWer:UL:PSRSoffset
```

class PsRsOffsetCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:UL:PSRSoffset
value: int = driver.configure.signaling.lte.cell.power.uplink.psRsOffset.
↪get(cell_name = 'abc')
```

Sets the parameter 'pSRS-Offset', signaled to the UE as an uplink power control parameter.

param cell_name
No help available

return
ps_rs_offset: No help available

set(*cell_name: str*, *ps_rs_offset: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:UL:PSRSoffset
driver.configure.signaling.lte.cell.power.uplink.psRsOffset.set(cell_name = 'abc
↪', ps_rs_offset = 1)
```

Sets the parameter 'pSRS-Offset', signaled to the UE as an uplink power control parameter.

param cell_name
No help available

param ps_rs_offset
No help available

6.3.4.7.2.95 Pucch

class PucchCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.lte.cell.power.uplink.pucch.clone()
```

Subgroups

6.3.4.7.2.96 Nominal

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:POWer:UL:PUCCh:NOMinal
```

class NominalCls

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

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:UL:PUCCh:NOMinal
value: float = driver.configure.signaling.lte.cell.power.uplink.pucch.nominal.
↪get(cell_name = 'abc')
```

Sets the UL power control parameter 'p0-NominalPUCCH'.

param cell_name
No help available

return
p_0_nominal_pucch: No help available

set(*cell_name: str*, *p_0_nominal_pucch: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:UL:PUCCh:NOMinal
driver.configure.signaling.lte.cell.power.uplink.pucch.nominal.set(cell_name =
↪'abc', p_0_nominal_pucch = 1.0)
```

Sets the UL power control parameter 'p0-NominalPUCCH'.

param cell_name
No help available

param p_0_nominal_pucch
No help available

6.3.4.7.2.97 Ue

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:POWer:UL:PUCCh:UE
```

class UeCls

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

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:UL:PUCCh:UE
value: float = driver.configure.signaling.lte.cell.power.uplink.pucch.ue.
↪get(cell_name = 'abc')
```

Sets the UL power control parameter 'p0-UE-PUCCH'.

param cell_name
No help available

return
p_0_ue_pucch: No help available

set(*cell_name: str*, *p_0_ue_pucch: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:UL:PUCCh:UE
driver.configure.signaling.lte.cell.power.uplink.pucch.ue.set(cell_name = 'abc',
↪ p_0_ue_pucch = 1.0)
```

Sets the UL power control parameter 'p0-UE-PUCCH'.

param cell_name
No help available

param p_0_ue_pucch
No help available

6.3.4.7.2.98 Pusch

class PuschCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.lte.cell.power.uplink.pusch.clone()
```

Subgroups

6.3.4.7.2.99 Nominal

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:POWer:UL:PUSCh:NOMinal
```

class NominalCls

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

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:UL:PUSCh:NOMinal
value: float = driver.configure.signaling.lte.cell.power.uplink.pusch.nominal.
↪get(cell_name = 'abc')
```

Sets the UL power control parameter 'p0-NominalPUSCH'.

param cell_name
No help available

return
p_0_nominal_pusch: No help available

set(*cell_name: str*, *p_0_nominal_pusch: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:UL:PUSCh:NOMinal
driver.configure.signaling.lte.cell.power.uplink.pusch.nominal.set(cell_name =
↪'abc', p_0_nominal_pusch = 1.0)
```

Sets the UL power control parameter 'p0-NominalPUSCH'.

param cell_name
No help available

param p_0_nominal_pusch
No help available

6.3.4.7.2.100 Ue

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:POWer:UL:PUSCh:UE
```

class UeCls

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

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:UL:PUSCh:UE
value: float = driver.configure.signaling.lte.cell.power.uplink.pusch.ue.
↪get(cell_name = 'abc')
```

Sets the UL power control parameter 'p0-UE-PUSCH'.

param cell_name
No help available

return
p_0_ue_pusch: No help available

set(*cell_name: str*, *p_0_ue_pusch: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:UL:PUSCh:UE
driver.configure.signaling.lte.cell.power.uplink.pusch.ue.set(cell_name = 'abc',
↪ p_0_ue_pusch = 1.0)
```

Sets the UL power control parameter 'p0-UE-PUSCH'.

param cell_name
No help available

param p_0_ue_pusch
No help available

6.3.4.7.2.101 Rms

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:POWer:UL:RMS
```

class RmsCls

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

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:UL:RMS
value: float = driver.configure.signaling.lte.cell.power.uplink.rms.get(cell_
↪name = 'abc')
```

Sets the maximum RMS power expected in the UL, for user-defined configuration. For automatic configuration, you can query the value.

param cell_name
No help available

return
max_exp_rms_pwr: No help available

set(*cell_name: str*, *max_exp_rms_pwr: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:UL:RMS
driver.configure.signaling.lte.cell.power.uplink.rms.set(cell_name = 'abc', max_
↪exp_rms_pwr = 1.0)
```

Sets the maximum RMS power expected in the UL, for user-defined configuration. For automatic configuration, you can query the value.

param cell_name
No help available

param max_exp_rms_pwr
No help available

6.3.4.7.2.102 ZczConfig

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:POWer:UL:ZCZConfig
```

class ZczConfigCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:UL:ZCZConfig
value: int = driver.configure.signaling.lte.cell.power.uplink.zczConfig.
↪get(cell_name = 'abc')
```

Sets the parameter 'zeroCorrelationZoneConfig', signaled to the UE.

param cell_name
No help available

return
config: No help available

set(*cell_name: str*, *config: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:POWer:UL:ZCZConfig
driver.configure.signaling.lte.cell.power.uplink.zczConfig.set(cell_name = 'abc
↪', config = 1)
```

Sets the parameter 'zeroCorrelationZoneConfig', signaled to the UE.

param cell_name
No help available

param config
No help available

6.3.4.7.2.103 Pusch

class PuschCls

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

Cloning the Group

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

Subgroups

6.3.4.7.2.104 Qam<QamOrder>

RepCap Settings

```
# Range: Order64 .. Order256
rc = driver.configure.signaling.lte.cell.pusch.qam.repcap_qamOrder_get()
driver.configure.signaling.lte.cell.pusch.qam.repcap_qamOrder_set(repcap.QamOrder.
↪Order64)
```

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:PUSCh:QAM<nr>
```

class QamCls

Qam commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: QamOrder, default value after init: QamOrder.Order64

get(*cell_name: str*, *qamOrder=QamOrder.Default*) → bool

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:PUSCh:QAM<nr>
value: bool = driver.configure.signaling.lte.cell.pusch.qam.get(cell_name = 'abc
↪', qamOrder = repcap.QamOrder.Default)
```

No command help available

param cell_name
No help available

param qamOrder
optional repeated capability selector. Default value: Order64 (settable in the interface 'Qam')

return
enable: No help available

set(*cell_name: str*, *enable: bool*, *qamOrder=QamOrder.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:PUSCh:QAM<nr>
driver.configure.signaling.lte.cell.pusch.qam.set(cell_name = 'abc', enable =␣
↪False, qamOrder = repcap.QamOrder.Default)
```

No command help available

param cell_name
No help available

param enable
No help available

param qamOrder
optional repeated capability selector. Default value: Order64 (settable in the interface 'Qam')

Cloning the Group

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

6.3.4.7.2.105 ReSelection

class ReSelectionCls

ReSelection commands group definition. 9 total commands, 6 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.4.7.2.106 Common

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:RESelection:COMMon
```

class CommonCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RESelection:COMMon
value: int = driver.configure.signaling.lte.cell.reSelection.common.get(cell_
↪name = 'abc')
```

Configures the parameter 'q-Hyst', signaled to the UE in SIB3.

param cell_name
No help available

return
power: No help available

set(*cell_name: str*, *power: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RESelection:COMMon
driver.configure.signaling.lte.cell.reSelection.common.set(cell_name = 'abc',␣
↪power = 1)
```

Configures the parameter 'q-Hyst', signaled to the UE in SIB3.

param cell_name
No help available

param power
No help available

6.3.4.7.2.107 MinLevel

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:RESelection:MINLevel
```

class MinLevelCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RESelection:MINLevel
value: int = driver.configure.signaling.lte.cell.reSelection.minLevel.get(cell_
↪name = 'abc')
```

Configures the parameter 'q-RxLevMin', signaled to the UE in SIB3.

param cell_name
No help available

return
power: No help available

set(*cell_name: str*, *power: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RESelection:MINLevel
driver.configure.signaling.lte.cell.reSelection.minLevel.set(cell_name = 'abc',␣
↪power = 1)
```

Configures the parameter 'q-RxLevMin', signaled to the UE in SIB3.

param cell_name
No help available

param power
No help available

6.3.4.7.2.108 Priority

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:RESelection:PRIority
```

class PriorityCls

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

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RESelection:PRIority
value: float = driver.configure.signaling.lte.cell.reSelection.priority.
↪get(cell_name = 'abc')
```

Configures the parameter 'cellReselectionPriority', signaled to the UE in SIB3.

param cell_name
No help available

return
priority: No help available

set(*cell_name: str*, *priority: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RESelection:PRIority
driver.configure.signaling.lte.cell.reSelection.priority.set(cell_name = 'abc',␣
↪priority = 1.0)
```

Configures the parameter 'cellReselectionPriority', signaled to the UE in SIB3.

param cell_name
No help available

param priority
No help available

6.3.4.7.2.109 Search

class SearchCls

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

Cloning the Group

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

Subgroups

6.3.4.7.2.110 Intrasearch

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:RESelection:SEARch:INTRasearch
```

class IntrasearchCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RESelection:SEARch:INTRasearch
value: int or bool = driver.configure.signaling.lte.cell.reSelection.search.
↪intrasearch.get(cell_name = 'abc')
```

Configures the parameter 's-IntraSearchP', signaled to the UE in SIB3 if the value is not OFF.

param cell_name
No help available

return
power: (integer or boolean) No help available

set(*cell_name: str*, *power: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RESelection:SEARch:INTRasearch
driver.configure.signaling.lte.cell.reSelection.search.intrasearch.set(cell_
↪name = 'abc', power = 1)
```

Configures the parameter 's-IntraSearchP', signaled to the UE in SIB3 if the value is not OFF.

param cell_name
No help available

param power
(integer or boolean) No help available

6.3.4.7.2.111 Nintrasearch

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:RESelection:SEARch:NINTrasearch
```

class NintrasearchCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RESelection:SEARch:NINTrasearch
value: int or bool = driver.configure.signaling.lte.cell.reSelection.search.
↪nintrasearch.get(cell_name = 'abc')
```

Configures the parameter 's-NonIntraSearchP', signaled to the UE in SIB3 if the value is not OFF.

param cell_name
No help available

return
power: (integer or boolean) No help available

set(*cell_name: str*, *power: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RESelection:SEARch:NINTrasearch
driver.configure.signaling.lte.cell.reSelection.search.nintrasearch.set(cell_
↪name = 'abc', power = 1)
```

Configures the parameter 's-NonIntraSearchP', signaled to the UE in SIB3 if the value is not OFF.

param cell_name
No help available

param power
(integer or boolean) No help available

6.3.4.7.2.112 Thresholds

class ThresholdsCls

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

Cloning the Group

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

Subgroups

6.3.4.7.2.113 HigHq

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:RESelection:THResholds:HIGHq
```

class HigHqCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RESelection:THResholds:HIGHq
value: int or bool = driver.configure.signaling.lte.cell.reSelection.thresholds.
↪higHq.get(cell_name = 'abc')
```

Configures the parameter 'threshX-HighQ', signaled to the UE in SIB5 if the value is not OFF.

param cell_name
No help available

return
power: (integer or boolean) No help available

set(*cell_name: str*, *power: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RESelection:THResholds:HIGHq
driver.configure.signaling.lte.cell.reSelection.thresholds.higHq.set(cell_name␣
↪= 'abc', power = 1)
```

Configures the parameter 'threshX-HighQ', signaled to the UE in SIB5 if the value is not OFF.

param cell_name
No help available

param power
(integer or boolean) No help available

6.3.4.7.2.114 Lowp

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:RESelection:THResholds:LOWP
```

class LowpCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RESelection:THResholds:LOWP
value: int = driver.configure.signaling.lte.cell.reSelection.thresholds.lowp.
↪get(cell_name = 'abc')
```

Configures the parameter 'threshServingLow', signaled to the UE in SIB3.

param cell_name
No help available

return
power: No help available

set(*cell_name: str*, *power: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RESelection:THResholds:LOWP
driver.configure.signaling.lte.cell.reSelection.thresholds.lowp.set(cell_name =
↪'abc', power = 1)
```

Configures the parameter 'threshServingLow', signaled to the UE in SIB3.

param cell_name
No help available

param power
No help available

6.3.4.7.2.115 Lowq

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:RESelection:THResholds:LOWQ
```

class LowqCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RESelection:THResholds:LOWQ
value: int or bool = driver.configure.signaling.lte.cell.reSelection.thresholds.
↪lowq.get(cell_name = 'abc')
```

Configures the parameters 'threshServingLowQ' and 'threshX-LowQ', signaled to the UE in SIB3 / SIB5 if the value is not OFF.

param cell_name
No help available

return
power: (integer or boolean) No help available

set(*cell_name: str*, *power: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RESelection:THResholds:LOWQ
driver.configure.signaling.lte.cell.reSelection.thresholds.lowq.set(cell_name =
↪'abc', power = 1)
```

Configures the parameters 'threshServingLowQ' and 'threshX-LowQ', signaled to the UE in SIB3 / SIB5 if the value is not OFF.

param cell_name
No help available

param power
(integer or boolean) No help available

6.3.4.7.2.116 Timer

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:RESelection:TIMer
```

class TimerCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RESelection:TIMer
value: int = driver.configure.signaling.lte.cell.reSelection.timer.get(cell_
↪name = 'abc')
```

Configures the parameter 't-ReselectionEUTRA', signaled to the UE in SIB3.

param cell_name
No help available

return
time: No help available

set(*cell_name: str*, *time: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RESelection:TIMer
driver.configure.signaling.lte.cell.reSelection.timer.set(cell_name = 'abc',␣
↪time = 1)
```

Configures the parameter 't-ReselectionEUTRA', signaled to the UE in SIB3.

param cell_name
No help available

param time
No help available

6.3.4.7.2.117 RfSettings

class RfSettingsCls

RfSettings commands group definition. 19 total commands, 8 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.4.7.2.118 AsEmission

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:RFSettings:ASEMission
```

class AsEmissionCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RFSettings:ASEMission
value: int or bool = driver.configure.signaling.lte.cell.rfSettings.asEmission.
↪get(cell_name = 'abc')
```

Sets the parameter 'AdditionalSpectrumEmission', signaled to the UE if the value is not OFF.

param cell_name
No help available

return
as_emission: (integer or boolean) No help available

set(*cell_name: str*, *as_emission: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RFSettings:ASEMission
driver.configure.signaling.lte.cell.rfSettings.asEmission.set(cell_name = 'abc',
↪ as_emission = 1)
```

Sets the parameter 'AdditionalSpectrumEmission', signaled to the UE if the value is not OFF.

param cell_name
No help available

param as_emission
(integer or boolean) No help available

6.3.4.7.2.119 Bchannel

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:RFSettings:BCHannel
```

class BchannelCls

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

class GetStruct

Response structure. Fields:

- Fbi: int: Frequency band indicator.
- Dl_Channel: int: No parameter help available
- Dl_Bandwidth: enums.UlBandwidth: Bxyz means xy.z MHz.
- Ul_Channel: int: No parameter help available
- Ul_Bandwidth: enums.UlBandwidth: Bxyz means xy.z MHz.

class SetStruct

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

- Cell_Name: str: No parameter help available
- Fbi: int: Frequency band indicator.
- Dl_Channel: int: No parameter help available
- Dl_Bandwidth: enums.UlBandwidth: Optional setting parameter. Bxyz means xy.z MHz.
- Ul_Channel: int: No parameter help available
- Ul_Bandwidth: enums.UlBandwidth: Optional setting parameter. Bxyz means xy.z MHz.

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RFSettings:BCHannel
value: GetStruct = driver.configure.signaling.lte.cell.rfSettings.bchannel.
↪get(cell_name = 'abc')
```

Defines the frequency band, the channel numbers and the bandwidths.

param cell_name
No help available

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

set(*structure: SetStruct*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RFSettings:BCHannel
structure = driver.configure.signaling.lte.cell.rfSettings.bchannel.SetStruct()
structure.Cell_Name: str = 'abc'
structure.Fbi: int = 1
structure.Dl_Channel: int = 1
structure.Dl_Bandwidth: enums.UlBandwidth = enums.UlBandwidth.B014
structure.Ul_Channel: int = 1
structure.Ul_Bandwidth: enums.UlBandwidth = enums.UlBandwidth.B014
driver.configure.signaling.lte.cell.rfSettings.bchannel.set(structure)
```

Defines the frequency band, the channel numbers and the bandwidths.

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

6.3.4.7.2.120 Combined

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:RFSettings:COMBined
```

class CombinedCls

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

class GetStruct

Response structure. Fields:

- Duplex_Mode: enums.DuplexModeB: No parameter help available
- Fbi: int: Frequency band indicator.
- Dl_Channel: int: No parameter help available
- Dl_Bandwidth: enums.UlBandwidth: Bxyz means xy.z MHz.
- Ul_Channel: int: No parameter help available
- Ul_Bandwidth: enums.UlBandwidth: Bxyz means xy.z MHz.

class SetStruct

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

- Cell_Name: str: No parameter help available
- Duplex_Mode: enums.DuplexModeB: No parameter help available
- Fbi: int: Optional setting parameter. Frequency band indicator.
- Dl_Channel: int: No parameter help available
- Dl_Bandwidth: enums.UlBandwidth: Optional setting parameter. Bxyz means xy.z MHz.
- Ul_Channel: int: No parameter help available
- Ul_Bandwidth: enums.UlBandwidth: Optional setting parameter. Bxyz means xy.z MHz.

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RFSettings:COMBined
value: GetStruct = driver.configure.signaling.lte.cell.rfSettings.combined.
↪get(cell_name = 'abc')
```

Defines the duplex mode, the frequency band, the channel numbers and the bandwidths.

param cell_name
No help available

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

set(*structure: SetStruct*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RFSettings:COMBined
structure = driver.configure.signaling.lte.cell.rfSettings.combined.SetStruct()
structure.Cell_Name: str = 'abc'
structure.Duplex_Mode: enums.DuplexModeB = enums.DuplexModeB.FDD
```

(continues on next page)

(continued from previous page)

```
structure.Fbi: int = 1
structure.Dl_Channel: int = 1
structure.Dl_Bandwidth: enums.UlBandwidth = enums.UlBandwidth.B014
structure.Ul_Channel: int = 1
structure.Ul_Bandwidth: enums.UlBandwidth = enums.UlBandwidth.B014
driver.configure.signaling.lte.cell.rfSettings.combined.set(structure)
```

Defines the duplex mode, the frequency band, the channel numbers and the bandwidths.

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

6.3.4.7.2.121 Dmode

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:RFSettings:DMODe
```

class DmodeCls

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

get(*cell_name: str*) → DuplexModeB

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RFSettings:DMODe
value: enums.DuplexModeB = driver.configure.signaling.lte.cell.rfSettings.dmode.
↪get(cell_name = 'abc')
```

Selects the duplex mode.

param cell_name
No help available

return
duplex_mode: No help available

set(*cell_name: str*, *duplex_mode: DuplexModeB*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RFSettings:DMODe
driver.configure.signaling.lte.cell.rfSettings.dmode.set(cell_name = 'abc',␣
↪duplex_mode = enums.DuplexModeB.FDD)
```

Selects the duplex mode.

param cell_name
No help available

param duplex_mode
No help available

6.3.4.7.2.122 Downlink

class DownlinkCls

Downlink commands group definition. 6 total commands, 6 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.4.7.2.123 Bandwidth

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:RFSettings:DL:BWIDth
```

class BandwidthCls

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

class GetStruct

Response structure. Fields:

- Bandwidth: enums.UlBandwidth: Bxyz means xy.z MHz.
- Resource_Blocks: int: Number of allocated resource blocks (full allocation of the bandwidth) .

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RFSettings:DL:BWIDth
value: GetStruct = driver.configure.signaling.lte.cell.rfSettings.downlink.
↪bandwidth.get(cell_name = 'abc')
```

Selects the channel bandwidth for the downlink. Configure the same value for DL and UL. A query returns <Bandwidth>, <ResourceBlocks>.

param cell_name
No help available

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

set(*cell_name: str*, *bandwidth: UlBandwidth*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RFSettings:DL:BWIDth
driver.configure.signaling.lte.cell.rfSettings.downlink.bandwidth.set(cell_name␣
↪= 'abc', bandwidth = enums.UlBandwidth.B014)
```

Selects the channel bandwidth for the downlink. Configure the same value for DL and UL. A query returns <Bandwidth>, <ResourceBlocks>.

param cell_name
No help available

param bandwidth
Bxyz means xy.z MHz.

6.3.4.7.2.124 Earfcn

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:RFSettings:DL:EARFcn
```

class EarfcnCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RFSettings:DL:EARFcn
value: int = driver.configure.signaling.lte.cell.rfSettings.downlink.earfcn.
↪get(cell_name = 'abc')
```

Selects the DL channel number. For FDD, the UL channel number is also set, using the default UL-DL separation.

param cell_name
No help available

return
channel: HASYmmetric refers to the high DL-only range, e.g. for band 66.

set(*cell_name: str*, *channel: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RFSettings:DL:EARFcn
driver.configure.signaling.lte.cell.rfSettings.downlink.earfcn.set(cell_name =
↪'abc', channel = 1)
```

Selects the DL channel number. For FDD, the UL channel number is also set, using the default UL-DL separation.

param cell_name
No help available

param channel
HASYmmetric refers to the high DL-only range, e.g. for band 66.

6.3.4.7.2.125 FreqError

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:RFSettings:DL:FERRor
```

class FreqErrorCls

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

class GetStruct

Response structure. Fields:

- Frequency_Error: float: No parameter help available

- Res_Center_Freq: int: Center frequency resulting from nominal center frequency plus frequency error.

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RFSettings:DL:FERRor
value: GetStruct = driver.configure.signaling.lte.cell.rfSettings.downlink.
↪freqError.get(cell_name = 'abc')
```

Configures a frequency error to be added to the configured DL carrier center frequency.

param cell_name
No help available

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

set(*cell_name: str*, *frequency_error: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RFSettings:DL:FERRor
driver.configure.signaling.lte.cell.rfSettings.downlink.freqError.set(cell_name␣
↪= 'abc', frequency_error = 1.0)
```

Configures a frequency error to be added to the configured DL carrier center frequency.

param cell_name
No help available

param frequency_error
No help available

6.3.4.7.2.126 Frequency

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:RFSettings:DL:FREQuency
```

class FrequencyCls

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

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RFSettings:DL:FREQuency
value: float = driver.configure.signaling.lte.cell.rfSettings.downlink.
↪frequency.get(cell_name = 'abc')
```

Defines the DL frequency. For FDD, the UL frequency is also set, using the default UL-DL separation.

param cell_name
No help available

return
frequency: No help available

set(*cell_name: str*, *frequency: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RFSettings:DL:FREQuency
driver.configure.signaling.lte.cell.rfSettings.downlink.frequency.set(cell_name␣
↪= 'abc', frequency = 1.0)
```

Defines the DL frequency. For FDD, the UL frequency is also set, using the default UL-DL separation.

param cell_name
No help available

param frequency
No help available

6.3.4.7.2.127 Rblocks

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:RFSettings:DL:RBLocks
```

class RblocksCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RFSettings:DL:RBLocks
value: int = driver.configure.signaling.lte.cell.rfSettings.downlink.rblocks.
↪get(cell_name = 'abc')
```

Selects the number of downlink RBs for full allocation and thus the downlink channel bandwidth. Configure the same value for DL and UL.

param cell_name
No help available

return
resource_blocks: No help available

set(*cell_name: str*, *resource_blocks: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RFSettings:DL:RBLocks
driver.configure.signaling.lte.cell.rfSettings.downlink.rblocks.set(cell_name =
↪'abc', resource_blocks = 1)
```

Selects the number of downlink RBs for full allocation and thus the downlink channel bandwidth. Configure the same value for DL and UL.

param cell_name
No help available

param resource_blocks
No help available

6.3.4.7.2.128 Rchoice

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:RFSettings:DL:RCHoice
```

class RchoiceCls

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

get(*cell_name: str*) → RangeChoice

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RFSettings:DL:RCHoice
value: enums.RangeChoice = driver.configure.signaling.lte.cell.rfSettings.
↪downlink.rchoice.get(cell_name = 'abc')
```

Selects a method for DL frequency configuration.

param cell_name
No help available

return
range_choice: HASYmmetric refers to the high DL-only range, e.g. for band 66.

set(*cell_name: str*, *range_choice: RangeChoice*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RFSettings:DL:RCHoice
driver.configure.signaling.lte.cell.rfSettings.downlink.rchoice.set(cell_name =
↪'abc', range_choice = enums.RangeChoice.HASYmmetric)
```

Selects a method for DL frequency configuration.

param cell_name
No help available

param range_choice
HASYmmetric refers to the high DL-only range, e.g. for band 66.

6.3.4.7.2.129 FbIndicator

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:RFSettings:FBINdicator
```

class FbIndicatorCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RFSettings:FBINdicator
value: int = driver.configure.signaling.lte.cell.rfSettings.fbIndicator.
↪get(cell_name = 'abc')
```

Defines the frequency band.

param cell_name
No help available

return
fbi: No help available

set(*cell_name: str*, *fbi: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RFSettings:FBINdicator
driver.configure.signaling.lte.cell.rfSettings.fbIndicator.set(cell_name = 'abc
↪', fbi = 1)
```

Defines the frequency band.

param cell_name
No help available

param fbi
No help available

6.3.4.7.2.130 RbMax

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:RFSettings:RBMax
```

class RbMaxCls

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

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RFSettings:RBMax
value: bool = driver.configure.signaling.lte.cell.rfSettings.rbMax.get(cell_
↪name = 'abc')
```

The command enables the automatic configuration of a full scheduled RB allocation when the frequency bandwidth is increased.

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RFSettings:RBMax
driver.configure.signaling.lte.cell.rfSettings.rbMax.set(cell_name = 'abc',␣
↪enable = False)
```

The command enables the automatic configuration of a full scheduled RB allocation when the frequency bandwidth is increased.

param cell_name
No help available

param enable
No help available

6.3.4.7.2.131 Uplink

class UplinkCls

Uplink commands group definition. 7 total commands, 7 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.4.7.2.132 Bandwidth

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:RFSettings:UL:BWIDth
```

class BandwidthCls

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

class GetStruct

Response structure. Fields:

- Bandwidth: enums.UlBandwidth: Bxyz means xy.z MHz.
- Resource_Blocks: int: Number of allocated resource blocks (full allocation of the bandwidth) .

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RFSettings:UL:BWIDth
value: GetStruct = driver.configure.signaling.lte.cell.rfSettings.uplink.
↪bandwidth.get(cell_name = 'abc')
```

Selects the channel bandwidth for the uplink. Configure the same value for DL and UL. A query returns <Bandwidth>, <ResourceBlocks>.

param cell_name
No help available

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

set(*cell_name: str*, *bandwidth: UlBandwidth*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RFSettings:UL:BWIDth
driver.configure.signaling.lte.cell.rfSettings.uplink.bandwidth.set(cell_name =
↪'abc', bandwidth = enums.UlBandwidth.B014)
```

Selects the channel bandwidth for the uplink. Configure the same value for DL and UL. A query returns <Bandwidth>, <ResourceBlocks>.

param cell_name
No help available

param bandwidth
Bxyz means xy.z MHz.

6.3.4.7.2.133 Earfcn

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:RFSettings:UL:EARFcn
```

class EarfcnCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RFSettings:UL:EARFcn
value: int = driver.configure.signaling.lte.cell.rfSettings.uplink.earfcn.
↪get(cell_name = 'abc')
```

Selects the UL channel number. The DL channel number is not changed.

param cell_name
No help available

return
channel: HASYmmetric refers to the high DL-only range, e.g. for band 66.

set(*cell_name: str*, *channel: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RFSettings:UL:EARFcn
driver.configure.signaling.lte.cell.rfSettings.uplink.earfcn.set(cell_name =
↪'abc', channel = 1)
```

Selects the UL channel number. The DL channel number is not changed.

param cell_name
No help available

param channel
HASYmmetric refers to the high DL-only range, e.g. for band 66.

6.3.4.7.2.134 FreqError

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:RFSettings:UL:FERRor
```

class FreqErrorCls

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

class GetStruct

Response structure. Fields:

- Frequency_Error: float: No parameter help available
- Res_Center_Freq: int: Center frequency resulting from nominal center frequency plus frequency error.

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RFSettings:UL:FERRor
value: GetStruct = driver.configure.signaling.lte.cell.rfSettings.uplink.
↪freqError.get(cell_name = 'abc')
```

Configures a frequency error to be added to the configured UL carrier center frequency.

param cell_name
No help available

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

set(*cell_name: str*, *frequency_error: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RFSettings:UL:FERRor
driver.configure.signaling.lte.cell.rfSettings.uplink.freqError.set(cell_name =
↪'abc', frequency_error = 1.0)
```

Configures a frequency error to be added to the configured UL carrier center frequency.

param cell_name
No help available

param frequency_error
No help available

6.3.4.7.2.135 Frequency

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:RFSettings:UL:FREQuency
```

class FrequencyCls

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

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RFSettings:UL:FREQuency
value: float = driver.configure.signaling.lte.cell.rfSettings.uplink.frequency.
↪get(cell_name = 'abc')
```

Defines the UL frequency. The DL frequency is not changed.

param cell_name
No help available

return
frequency: No help available

set(*cell_name: str*, *frequency: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RFSettings:UL:FREQuency
driver.configure.signaling.lte.cell.rfSettings.uplink.frequency.set(cell_name =
↪'abc', frequency = 1.0)
```

Defines the UL frequency. The DL frequency is not changed.

param cell_name
No help available

param frequency
No help available

6.3.4.7.2.136 Rblocks

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:RFSettings:UL:RBLocks
```

class RblocksCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RFSettings:UL:RBLocks
value: int = driver.configure.signaling.lte.cell.rfSettings.uplink.rblocks.
↪get(cell_name = 'abc')
```

Selects the number of uplink RBs for full allocation and thus the uplink channel bandwidth. Configure the same value for DL and UL.

param cell_name
No help available

return
resource_blocks: No help available

set(*cell_name: str*, *resource_blocks: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RFSettings:UL:RBLocks
driver.configure.signaling.lte.cell.rfSettings.uplink.rblocks.set(cell_name =
↪'abc', resource_blocks = 1)
```

Selects the number of uplink RBs for full allocation and thus the uplink channel bandwidth. Configure the same value for DL and UL.

param cell_name
No help available

param resource_blocks
No help available

6.3.4.7.2.137 Rchoice

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:RFSettings:UL:RCHoice
```

class RchoiceCls

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

get(*cell_name: str*) → RangeChoice

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RFSettings:UL:RCHoice
value: enums.RangeChoice = driver.configure.signaling.lte.cell.rfSettings.
↪uplink.rchoice.get(cell_name = 'abc')
```

Selects a method for UL frequency configuration.

param cell_name
No help available

return
range_choice: HASYmmetric refers to the high DL-only range, e.g. for band 66.

set(*cell_name: str*, *range_choice: RangeChoice*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RFSettings:UL:RCHoice
driver.configure.signaling.lte.cell.rfSettings.uplink.rchoice.set(cell_name =
↪'abc', range_choice = enums.RangeChoice.HASYmmetric)
```

Selects a method for UL frequency configuration.

param cell_name
No help available

param range_choice
HASYmmetric refers to the high DL-only range, e.g. for band 66.

6.3.4.7.2.138 TxRx

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:RFSettings:UL:TXRX
```

class TxRxCls

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

get(*cell_name: str*) → TxRxSeparation

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RFSettings:UL:TXRX
value: enums.TxRxSeparation = driver.configure.signaling.lte.cell.rfSettings.
↪uplink.txRx.get(cell_name = 'abc')
```

Selects a configuration method for the uplink carrier center frequency, for FDD.

param cell_name
No help available

return
tx_rx_separation: UDEFined: Define UL frequency independent of DL frequency. DEFault: Use standardized UL-DL separation.

set(*cell_name: str*, *tx_rx_separation: TxRxSeparation*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:RFSettings:UL:TXRX
driver.configure.signaling.lte.cell.rfSettings.uplink.txRx.set(cell_name = 'abc
↪', tx_rx_separation = enums.TxRxSeparation.DEFault)
```

Selects a configuration method for the uplink carrier center frequency, for FDD.

param cell_name
No help available

param tx_rx_separation
UDEFined: Define UL frequency independent of DL frequency. DEFault: Use standardized UL-DL separation.

6.3.4.7.2.139 Srs

class SrsCls

Srs commands group definition. 8 total commands, 3 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.4.7.2.140 Common

class CommonCls

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

Cloning the Group

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

Subgroups

6.3.4.7.2.141 Bandwidth

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:SRS:COMMon:BWIDth
```

class BandwidthCls

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

get(*cell_name: str*) → BandwidthCommon

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:SRS:COMMon:BWIDth
value: enums.BandwidthCommon = driver.configure.signaling.lte.cell.srs.common.
↪bandwidth.get(cell_name = 'abc')
```

Configures the parameter 'srs-BandwidthConfig'. Only configurable for the mode UDEFined.

param cell_name
No help available

return
bandwidth: No help available

set(*cell_name: str*, *bandwidth: BandwidthCommon*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:SRS:COMMon:BWIDth
driver.configure.signaling.lte.cell.srs.common.bandwidth.set(cell_name = 'abc',␣
↪bandwidth = enums.BandwidthCommon.BW0)
```

Configures the parameter 'srs-BandwidthConfig'. Only configurable for the mode UDEFined.

param cell_name
No help available

param bandwidth
No help available

6.3.4.7.2.142 Sant

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:SRS:COMMon:SANT
```

class SantCls

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

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:SRS:COMMon:SANT
value: bool = driver.configure.signaling.lte.cell.srs.common.sant.get(cell_name␣
↪= 'abc')
```

Configures the parameter 'ackNackSRS-SimultaneousTransmission'. Only configurable for the mode UDEFined.

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:SRS:COMMon:SANT
driver.configure.signaling.lte.cell.srs.common.sant.set(cell_name = 'abc',␣
↪enable = False)
```

Configures the parameter 'ackNackSRS-SimultaneousTransmission'. Only configurable for the mode UDEFined.

param cell_name
No help available

param enable
No help available

6.3.4.7.2.143 Sframe

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:SRS:COMMon:SFRame
```

class SframeCls

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

get(*cell_name: str*) → Subframe

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:SRS:COMMon:SFRame
value: enums.Subframe = driver.configure.signaling.lte.cell.srs.common.sframe.
↪get(cell_name = 'abc')
```

Configures the parameter 'srs-SubframeConfig'. Only configurable for the mode UDEFined.

param cell_name
No help available

return
subframe: No help available

set(*cell_name: str*, *subframe: Subframe*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:SRS:COMMon:SFRame
driver.configure.signaling.lte.cell.srs.common.sframe.set(cell_name = 'abc',␣
↪subframe = enums.Subframe.SC0)
```

Configures the parameter 'srs-SubframeConfig'. Only configurable for the mode UDEFined.

param cell_name
No help available

param subframe
No help available

6.3.4.7.2.144 Dedicated

class DedicatedCls

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

Cloning the Group

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

Subgroups

6.3.4.7.2.145 Bandwidth

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:SRS:DEDicated:BWIDth
```

class BandwidthCls

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

get(*cell_name: str*) → BandwidthDedicated

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:SRS:DEDicated:BWIDth
value: enums.BandwidthDedicated = driver.configure.signaling.lte.cell.srs.
↪dedicated.bandwidth.get(cell_name = 'abc')
```

Configures the parameter 'srs-Bandwidth'. Only configurable for the mode UDEFined.

param cell_name
No help available

return
bandwidth: No help available

set(*cell_name: str*, *bandwidth: BandwidthDedicated*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:SRS:DEDicated:BWIDth
driver.configure.signaling.lte.cell.srs.dedicated.bandwidth.set(cell_name = 'abc
↪', bandwidth = enums.BandwidthDedicated.BW0)
```

Configures the parameter 'srs-Bandwidth'. Only configurable for the mode UDEFined.

param cell_name
No help available

param bandwidth
No help available

6.3.4.7.2.146 Cindex

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:SRS:DEDicated:CINDex
```

class CindexCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:SRS:DEDicated:CINDex
value: int = driver.configure.signaling.lte.cell.srs.dedicated.cindex.get(cell_
↪name = 'abc')
```

Configures the parameter 'srs-ConfigIndex'. Only configurable for the mode UDEFined.

param cell_name
No help available

return
index: No help available

set(*cell_name: str*, *index: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:SRS:DEDicated:CINDex
driver.configure.signaling.lte.cell.srs.dedicated.cindex.set(cell_name = 'abc',␣
↪index = 1)
```

Configures the parameter 'srs-ConfigIndex'. Only configurable for the mode UDEFined.

param cell_name
No help available

param index
No help available

6.3.4.7.2.147 Enable

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:SRS:DEDicated:ENABle
```

class EnableCls

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

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:SRS:DEDicated:ENABle
value: bool = driver.configure.signaling.lte.cell.srs.dedicated.enable.get(cell_
↪name = 'abc')
```

Enables sending of the dedicated SRS parameters to the UE. Only configurable for the mode UDEFined.

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:SRS:DEDicated:ENABle
driver.configure.signaling.lte.cell.srs.dedicated.enable.set(cell_name = 'abc',␣
↪enable = False)
```

Enables sending of the dedicated SRS parameters to the UE. Only configurable for the mode UDEFined.

param cell_name
No help available

param enable
No help available

6.3.4.7.2.148 HbWidth

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:SRS:DEDicated:HBWidth
```

class HbWidthCls

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

get(*cell_name: str*) → BandwidthHoping

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:SRS:DEDicated:HBWidth
value: enums.BandwidthHoping = driver.configure.signaling.lte.cell.srs.
↪dedicated.hbWidth.get(cell_name = 'abc')
```

Configures the parameter 'srs-HoppingBandwidth'. Only configurable for the mode UDEFined.

param cell_name
No help available

return
bandwidth: No help available

set(*cell_name: str*, *bandwidth: BandwidthHoping*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:SRS:DEDicated:HBWidth
driver.configure.signaling.lte.cell.srs.dedicated.hbWidth.set(cell_name = 'abc',
↪ bandwidth = enums.BandwidthHoping.HBW0)
```

Configures the parameter 'srs-HoppingBandwidth'. Only configurable for the mode UDEFined.

param cell_name
No help available

param bandwidth
No help available

6.3.4.7.2.149 Mode

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:SRS:MODE
```

class ModeCls

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

get(*cell_name: str*) → ModeSrs

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:SRS:MODE
value: enums.ModeSrs = driver.configure.signaling.lte.cell.srs.mode.get(cell_
↪name = 'abc')
```

Selects whether SRS is supported by the cell and via which method the signaled SRS parameters are configured.

param cell_name
No help available

return
mode: OFF: no SRS parameters signaled to UE UDEFined: configuration of SRS parameters via other commands in this chapter A508: automatic configuration according to 3GPP TS 36.508 A521: automatic configuration according to 3GPP TS 36.521

set(*cell_name: str*, *mode: ModeSrs*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:SRS:MODE
driver.configure.signaling.lte.cell.srs.mode.set(cell_name = 'abc', mode =␣
↪enums.ModeSrs.A508)
```

Selects whether SRS is supported by the cell and via which method the signaled SRS parameters are configured.

param cell_name
No help available

param mode
OFF: no SRS parameters signaled to UE UDEFined: configuration of SRS parameters via other commands in this chapter A508: automatic configuration according to 3GPP TS 36.508 A521: automatic configuration according to 3GPP TS 36.521

6.3.4.7.2.150 Tadvance

class TadvanceCls

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

Cloning the Group

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

Subgroups

6.3.4.7.2.151 Periodicity

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:TADVance:PERiodicity
```

class PeriodicityCls

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

get(*cell_name: str*) → TadvPeriodicity

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:TADVance:PERiodicity
value: enums.TadvPeriodicity = driver.configure.signaling.lte.cell.tadvance.
↪periodicity.get(cell_name = 'abc')
```

Configures the periodicity for sending timing advance commands to the UE.

param cell_name
No help available

return
periodicity: No help available

set(*cell_name: str*, *periodicity: TadvPeriodicity*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:TADVance:PERiodicity
driver.configure.signaling.lte.cell.tadvance.periodicity.set(cell_name = 'abc',␣
↪periodicity = enums.TadvPeriodicity.CONTinuous)
```

Configures the periodicity for sending timing advance commands to the UE.

param cell_name
No help available

param periodicity
No help available

6.3.4.7.2.152 Timing

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:TADVance:TIMing
```

class TimingCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:TADVance:TIMing
value: int = driver.configure.signaling.lte.cell.tadvance.timing.get(cell_name␣
↪= 'abc')
```

Configures a timing advance value to be sent to the UE.

param cell_name
No help available

return
ta: No help available

set(*cell_name: str*, *ta: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:TADVance:TIMing
driver.configure.signaling.lte.cell.tadvance.timing.set(cell_name = 'abc', ta =␣
↪1)
```

Configures a timing advance value to be sent to the UE.

param cell_name
No help available

param ta
No help available

6.3.4.7.2.153 Tdd

class TddCls

Tdd commands group definition. 3 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.4.7.2.154 Subframe

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:TDD:SUBFrame
```

class SubframeCls

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

class GetStruct

Response structure. Fields:

- Assignment: enums.Assignment: Subframe assignment, defining the combination of UL, DL and special subframes.
- Special_Pattern: enums.SpecialPattern: Inner structure of special subframes.

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:TDD:SUBFrame
value: GetStruct = driver.configure.signaling.lte.cell.tdd.subframe.get(cell_
↪name = 'abc')
```

Defines the structure of a TDD radio frame.

param cell_name
No help available

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

set(*cell_name: str*, *assignment: Assignment*, *special_pattern: SpecialPattern = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:TDD:SUBFrame
driver.configure.signaling.lte.cell.tdd.subframe.set(cell_name = 'abc',␣
↪assignment = enums.Assignment.NONE, special_pattern = enums.SpecialPattern.P0)
```

Defines the structure of a TDD radio frame.

param cell_name
No help available

param assignment
Subframe assignment, defining the combination of UL, DL and special subframes.

param special_pattern
Inner structure of special subframes.

Cloning the Group

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

Subgroups

6.3.4.7.2.155 Assignment

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:TDD:SUBFrame:ASSignment
```

class AssignmentCls

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

get(*cell_name: str*) → Assignment

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:TDD:SUBFrame:ASSignment
value: enums.Assignment = driver.configure.signaling.lte.cell.tdd.subframe.
↪assignment.get(cell_name = 'abc')
```

Selects the subframe assignment, defining the combination of UL, DL and special subframes within a TDD radio frame.

param cell_name
No help available

return
assignment: No help available

set(*cell_name: str*, *assignment: Assignment*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:TDD:SUBFrame:ASSignment
driver.configure.signaling.lte.cell.tdd.subframe.assignment.set(cell_name = 'abc
↪', assignment = enums.Assignment.NONE)
```

Selects the subframe assignment, defining the combination of UL, DL and special subframes within a TDD radio frame.

param cell_name
No help available

param assignment
No help available

6.3.4.7.2.156 Special

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:TDD:SUBFrame:SPECial
```

class SpecialCls

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

get(*cell_name: str*) → SpecialPattern

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:TDD:SUBFrame:SPECial
value: enums.SpecialPattern = driver.configure.signaling.lte.cell.tdd.subframe.
↪special.get(cell_name = 'abc')
```

Selects a special subframe pattern (SSP) for TDD, defining the inner structure of special subframes.

param cell_name
No help available

return
special_pattern: P0 to P8: SSP 0 to SSP 8 P9: SSP 9 V1130 PAV1: SSP 10 V1430 PAV2: SSP 10 V1450 CRS Less DwPTS

set(*cell_name: str*, *special_pattern: SpecialPattern*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:TDD:SUBFrame:SPECial
driver.configure.signaling.lte.cell.tdd.subframe.special.set(cell_name = 'abc',␣
↪special_pattern = enums.SpecialPattern.P0)
```

Selects a special subframe pattern (SSP) for TDD, defining the inner structure of special subframes.

param cell_name
No help available

param special_pattern
P0 to P8: SSP 0 to SSP 8 P9: SSP 9 V1130 PAV1: SSP 10 V1430 PAV2: SSP 10 V1450 CRS Less DwPTS

6.3.4.7.2.157 Timeout

class TimeoutCls

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

Cloning the Group

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

Subgroups

6.3.4.7.2.158 N<Nnum>

RepCap Settings

```
# Range: Nr310 .. Nr311
rc = driver.configure.signaling.lte.cell.timeout.n.repcap_nnum_get()
driver.configure.signaling.lte.cell.timeout.n.repcap_nnum_set(repcap.Nnum.Nr310)
```

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:TOUT:N<no>
```

class NCls

N commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Nnum, default value after init: Nnum.Nr310

get(*cell_name: str*, *nnum=Nnum.Default*) → Counter

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:TOUT:N<no>
value: enums.Counter = driver.configure.signaling.lte.cell.timeout.n.get(cell_
↪name = 'abc', nnum = repcap.Nnum.Default)
```

(continues on next page)

(continued from previous page)

```
    INTRO_CMD_HELP: Configures one of the following constants:

    - N310, for counting out-of-sync indications and starting T310
    - N311, for counting in-sync indications and stopping T310

    :param cell_name: No help available
    :param nnum: optional repeated capability selector. Default value: Nr310␣
↪(settable in the interface 'N')
    :return: counter: Value of the constant For N310, the value N5 is not␣
↪allowed. For N311, the value N20 is not allowed.
```

set(*cell_name: str*, *counter: Counter*, *nnum=Nnum.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:TOUT:N<no>
driver.configure.signaling.lte.cell.timeout.n.set(cell_name = 'abc', counter =␣
↪enums.Counter.N1, nnum = repcap.Nnum.Default)


    INTRO_CMD_HELP: Configures one of the following constants:

    - N310, for counting out-of-sync indications and starting T310
    - N311, for counting in-sync indications and stopping T310

    :param cell_name: No help available
    :param counter: Value of the constant For N310, the value N5 is not allowed.
↪ For N311, the value N20 is not allowed.
    :param nnum: optional repeated capability selector. Default value: Nr310␣
↪(settable in the interface 'N')
```

Cloning the Group

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

6.3.4.7.2.159 T<Tnum>

RepCap Settings

```
# Range: Nr300 .. Nr319
rc = driver.configure.signaling.lte.cell.timeout.t.repcap_tnum_get()
driver.configure.signaling.lte.cell.timeout.t.repcap_tnum_set(repcap.Tnum.Nr300)
```

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:TOUT:T<no>
```

class TCls

T commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Tnum, default value after init: Tnum.Nr300

get(*cell_name: str*, *tnum=Tnum.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:TOUT:T<no>
value: int = driver.configure.signaling.lte.cell.timeout.t.get(cell_name = 'abc
↪', tnum = repcap.Tnum.Default)


    INTRO_CMD_HELP: Configures one of the following timers:

    - T300, RRC connection establishment
    - T301, RRC connection re-establishment, after cell selection
    - T310, detection of radio link failure (out-of-sync)
    - T311, RRC connection re-establishment, before cell selection

    :param cell_name: No help available
    :param tnum: optional repeated capability selector. Default value: Nr300␣
↪(settable in the interface 'T')
    :return: timer: Timeout value
```

set(*cell_name: str*, *timer: int*, *tnum=Tnum.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:TOUT:T<no>
driver.configure.signaling.lte.cell.timeout.t.set(cell_name = 'abc', timer = 1,␣
↪tnum = repcap.Tnum.Default)


    INTRO_CMD_HELP: Configures one of the following timers:

    - T300, RRC connection establishment
    - T301, RRC connection re-establishment, after cell selection
    - T310, detection of radio link failure (out-of-sync)
    - T311, RRC connection re-establishment, before cell selection

    :param cell_name: No help available
    :param timer: Timeout value
    :param tnum: optional repeated capability selector. Default value: Nr300␣
↪(settable in the interface 'T')
```

Cloning the Group

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

6.3.4.7.2.160 Timing

class TimingCls

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

Cloning the Group

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

Subgroups

6.3.4.7.2.161 DltShift

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:TIMing:DLTShift
```

class DltShiftCls

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

class GetStruct

Response structure. Fields:

- Delta: int: Offset in Ts
- Total: int: Total accumulated offsets

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:TIMing:DLTShift
value: GetStruct = driver.configure.signaling.lte.cell.timing.dltShift.get(cell_
↪name = 'abc')
```

Adds an offset to the DL timing, relative to the current timing. The setting is only configurable during a connection. Each time you change the setting, the new value is added to the previous timing. So the values accumulate. A query returns the last setting and the total accumulated value: <Delta>, <Total>.

param cell_name
No help available

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

set(*cell_name: str*, *delta: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:TIMing:DLTShift
driver.configure.signaling.lte.cell.timing.dltShift.set(cell_name = 'abc',␣
↪delta = 1)
```

Adds an offset to the DL timing, relative to the current timing. The setting is only configurable during a connection. Each time you change the setting, the new value is added to the previous timing. So the values accumulate. A query returns the last setting and the total accumulated value: <Delta>, <Total>.

param cell_name
No help available

param delta
Offset in Ts

6.3.4.7.2.162 Offset

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:TIMing:OFFSet
```

class OffsetCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:TIMing:OFFSet
value: int = driver.configure.signaling.lte.cell.timing.offset.get(cell_name =
↪'abc')
```

Defines a cell time offset.

param cell_name
No help available

return
time_offset: No help available

set(*cell_name: str*, *time_offset: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:TIMing:OFFSet
driver.configure.signaling.lte.cell.timing.offset.set(cell_name = 'abc', time_
↪offset = 1)
```

Defines a cell time offset.

param cell_name
No help available

param time_offset
No help available

6.3.4.7.2.163 SfnOffset

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:TIMing:SFNoffset
```

class SfnOffsetCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:TIMing:SFNoffset
value: int = driver.configure.signaling.lte.cell.timing.sfnOffset.get(cell_name␣
↪= 'abc')
```

Defines a system frame number offset.

param cell_name
No help available

return
sfn_offset: No help available

set(*cell_name: str*, *sfn_offset: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:TIMing:SFNoffset
driver.configure.signaling.lte.cell.timing.sfnOffset.set(cell_name = 'abc', sfn_
↪offset = 1)
```

Defines a system frame number offset.

param cell_name
No help available

param sfn_offset
No help available

6.3.4.7.2.164 UeScheduling

class UeSchedulingCls

UeScheduling commands group definition. 86 total commands, 8 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.4.7.2.165 CmMapping

class CmMappingCls

CmMapping commands group definition. 6 total commands, 3 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.4.7.2.166 Csirs

class CsirsCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.lte.cell.ueScheduling.cmMapping.csirs.clone()
```

Subgroups

6.3.4.7.2.167 Mcs

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:CMMapping:CSIRs:MCS
```

class McsCls

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

get(*cell_name: str*) → List[int]

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:CMMapping:CSIRs:MCS
value: List[int] = driver.configure.signaling.lte.cell.ueScheduling.cmMapping.
↪csirs.mcs.get(cell_name = 'abc')
```

Sets the configuration mode to UDEFined and defines the mapping table for that mode. A query returns the mapping table contents for the currently used configuration mode, without changing the mode. For setting the configuration mode, see [CONFigure:]SIGNaling:LTE:CELL:UEScheduling:CMMapping:CSIRs:MCSTable. There is a configuration mode and a mapping table for each type of DL subframe: CSI-RS subframe (CSIRs) , special subframe for TDD (SSUBframe) , all other subframes (NSUBframe) .

param cell_name
No help available

return
mcs: Comma-separated list of 16 MCS values, for reported CQI values 0 to 15.

set(*cell_name: str*, *mcs: List[int]*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:CMMapping:CSIRs:MCS
driver.configure.signaling.lte.cell.ueScheduling.cmMapping.csirs.mcs.set(cell_
↪name = 'abc', mcs = [1, 2, 3])
```

Sets the configuration mode to UDEFined and defines the mapping table for that mode. A query returns the mapping table contents for the currently used configuration mode, without changing the mode. For setting the configuration mode, see [CONFigure:]SIGNaling:LTE:CELL:UEScheduling:CMMapping:CSIRs:MCSTable. There is a configuration mode and a mapping table for each type of DL subframe: CSI-RS subframe (CSIRs) , special subframe for TDD (SSUBframe) , all other subframes (NSUBframe) .

param cell_name
No help available

param mcs
Comma-separated list of 16 MCS values, for reported CQI values 0 to 15.

6.3.4.7.2.168 McsTable

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:CMMapping:CSIRs:MCSTable
```

class McsTableCls

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

class GetStruct

Response structure. Fields:

- **Mcs_Table: enums.McsTableC:**
  - AUTO: Mapping table selected automatically, depending on [CONFigure:]SIGNaling:LTE:CELL:UEScheduling:UDEFined:SASSignment:DL:MCSTable.
  - P521: Mapping table contents defined by 3GPP TS 36.521.MCS scheme selection via Predefined3GPP.
  - UDEFined: Mapping table contents defined via separate command, see [CONFigure:]SIGNaling:LTE:CELL:UEScheduling:CMMapping:CSIRs:MCS etc.
- Predefined_3_Gpp: enums.Predefined3Gpp: Selects an MCS scheme for MCSTable = P521. Mn means 'MCS.n' in the tables A.

4-13 to A.4-16 in 3GPP TS 36.521.

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:CMMapping:CSIRs:MCSTable
value: GetStruct = driver.configure.signaling.lte.cell.ueScheduling.cmMapping.
↪csirs.mcsTable.get(cell_name = 'abc')
```

Selects a configuration mode for the CQI-MCS mapping tables for follow WB CQI. There is a mapping table for each type of DL subframe: CSI-RS subframe (CSIRs) , special subframe for TDD (SSUBframe) , all other subframes (NSUBframe) .

param cell_name
No help available

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

set(*cell_name: str*, *mcs_table: McsTableC*, *predefined_3_gpp: Predefined3Gpp = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:CMMapping:CSIRs:MCSTable
driver.configure.signaling.lte.cell.ueScheduling.cmMapping.csirs.mcsTable.
↪set(cell_name = 'abc', mcs_table = enums.McsTableC.AUTO, predefined_3_gpp =␣
↪enums.Predefined3Gpp.M1)
```

Selects a configuration mode for the CQI-MCS mapping tables for follow WB CQI. There is a mapping table for each type of DL subframe: CSI-RS subframe (CSIRs) , special subframe for TDD (SSUBframe) , all other subframes (NSUBframe) .

param cell_name
No help available

param mcs_table

- AUTO: Mapping table selected automatically, depending on [CONFigure:]SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:DL:MCSTable.
- P521: Mapping table contents defined by 3GPP TS 36.521.MCS scheme selection via Predefined3GPP.
- UDEFined: Mapping table contents defined via separate command, see [CONFigure:]SIGNaling:LTE:CELL:UESCheduling:CMMapping:CSIRs:MCS etc.

param predefined_3_gpp
Selects an MCS scheme for MCSTable = P521. Mn means 'MCS.n' in the tables A.4-13 to A.4-16 in

3GPP TS 36.521.

6.3.4.7.2.169 Nsubframe

class NsubframeCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.lte.cell.ueScheduling.cmMapping.nsubframe.clone()
```

Subgroups

6.3.4.7.2.170 Mcs

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:CMMapping:NSUBframe:MCS
```

class McsCls

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

get(*cell_name: str*) → List[int]

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:CMMapping:NSUBframe:MCS
value: List[int] = driver.configure.signaling.lte.cell.ueScheduling.cmMapping.
↪nsubframe.mcs.get(cell_name = 'abc')
```

Sets the configuration mode to UDEFined and defines the mapping table for that mode. A query returns the mapping table contents for the currently used configuration mode, without changing the mode. For setting the configuration mode, see [CONFigure:]SIGNaling:LTE:CELL:UEScheduling:CMMapping:CSIRs:MCSTable. There is a configuration mode and a mapping table for each type of DL subframe: CSI-RS subframe (CSIRs) , special subframe for TDD (SSUBframe) , all other subframes (NSUBframe) .

param cell_name
No help available

return
mcs: Comma-separated list of 16 MCS values, for reported CQI values 0 to 15.

set(*cell_name: str*, *mcs: List[int]*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:CMMapping:NSUBframe:MCS
driver.configure.signaling.lte.cell.ueScheduling.cmMapping.nsubframe.mcs.
↪set(cell_name = 'abc', mcs = [1, 2, 3])
```

Sets the configuration mode to UDEFined and defines the mapping table for that mode. A query returns the mapping table contents for the currently used configuration mode, without changing the mode. For setting the configuration mode, see [CONFigure:]SIGNaling:LTE:CELL:UEScheduling:CMMapping:CSIRs:MCSTable. There is a configuration mode and a mapping table for each type of DL subframe: CSI-RS subframe (CSIRs) , special subframe for TDD (SSUBframe) , all other subframes (NSUBframe) .

param cell_name
No help available

param mcs
Comma-separated list of 16 MCS values, for reported CQI values 0 to 15.

6.3.4.7.2.171 McsTable

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:CMMapping:NSUBframe:MCSTable
```

class McsTableCls

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

class GetStruct

Response structure. Fields:

- **Mcs_Table: enums.McsTableC:**
  - AUTO: Mapping table selected automatically, depending on [CONFigure:]SIGNaling:LTE:CELL:UEScheduling:UDEFined:SASSignment:DL:MCSTable.
  - P521: Mapping table contents defined by 3GPP TS 36.521.MCS scheme selection via Predefined3GPP.
  - UDEFined: Mapping table contents defined via separate command, see [CONFigure:]SIGNaling:LTE:CELL:UEScheduling:CMMapping:CSIRs:MCS etc.
- Predefined_3_Gpp: enums.Predefined3Gpp: Selects an MCS scheme for MCSTable = P521. Mn means 'MCS.n' in the tables A.

4-13 to A.4-16 in 3GPP TS 36.521.

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:CMMapping:NSUBframe:MCSTable
value: GetStruct = driver.configure.signaling.lte.cell.ueScheduling.cmMapping.
↪nsubframe.mcsTable.get(cell_name = 'abc')
```

Selects a configuration mode for the CQI-MCS mapping tables for follow WB CQI. There is a mapping table for each type of DL subframe: CSI-RS subframe (CSIRs) , special subframe for TDD (SSUBframe) , all other subframes (NSUBframe) .

param cell_name
No help available

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

set(*cell_name: str*, *mcs_table: McsTableC*, *predefined_3_gpp: Predefined3Gpp = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:CMMapping:NSUBframe:MCSTable
driver.configure.signaling.lte.cell.ueScheduling.cmMapping.nsubframe.mcsTable.
↪set(cell_name = 'abc', mcs_table = enums.McsTableC.AUTO, predefined_3_gpp =␣
↪enums.Predefined3Gpp.M1)
```

Selects a configuration mode for the CQI-MCS mapping tables for follow WB CQI. There is a mapping table for each type of DL subframe: CSI-RS subframe (CSIRs) , special subframe for TDD (SSUBframe) , all other subframes (NSUBframe) .

param cell_name
No help available

param mcs_table

- AUTO: Mapping table selected automatically, depending on [CONFigure:]SIGNaling:LTE:CELL:UEScheduling:UDEFined:SASSignment:DL:MCSTable.
- P521: Mapping table contents defined by 3GPP TS 36.521.MCS scheme selection via Predefined3GPP.
- UDEFined: Mapping table contents defined via separate command, see [CONFigure:]SIGNaling:LTE:CELL:UEScheduling:CMMapping:CSIRs:MCS etc.

param predefined_3_gpp
Selects an MCS scheme for MCSTable = P521. Mn means 'MCS.n' in the tables A.4-13 to A.4-16 in

3GPP TS 36.521.

6.3.4.7.2.172 Ssubframe

class SsubframeCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.lte.cell.ueScheduling.cmMapping.ssubframe.clone()
```

Subgroups

6.3.4.7.2.173 Mcs

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:CMMapping:SSUBframe:MCS
```

class McsCls

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

get(*cell_name: str*) → List[int]

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:CMMapping:SSUBframe:MCS
value: List[int] = driver.configure.signaling.lte.cell.ueScheduling.cmMapping.
↪ssubframe.mcs.get(cell_name = 'abc')
```

Sets the configuration mode to UDEFined and defines the mapping table for that mode. A query returns the mapping table contents for the currently used configuration mode, without changing the mode. For setting the configuration mode, see [CONFigure:]SIGNaling:LTE:CELL:UEScheduling:CMMapping:CSIRs:MCSTable. There is a configuration mode and a mapping table for each type of DL subframe: CSI-RS subframe (CSIRs) , special subframe for TDD (SSUBframe) , all other subframes (NSUBframe) .

param cell_name
No help available

return
mcs: Comma-separated list of 16 MCS values, for reported CQI values 0 to 15.

set(*cell_name: str*, *mcs: List[int]*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:CMMapping:SSUBframe:MCS
driver.configure.signaling.lte.cell.ueScheduling.cmMapping.ssubframe.mcs.
↪set(cell_name = 'abc', mcs = [1, 2, 3])
```

Sets the configuration mode to UDEFined and defines the mapping table for that mode. A query returns the mapping table contents for the currently used configuration mode, without changing the mode. For setting the configuration mode, see [CONFigure:]SIGNaling:LTE:CELL:UEScheduling:CMMapping:CSIRs:MCSTable. There is a configuration mode and a mapping table for each type of DL subframe: CSI-RS subframe (CSIRs) , special subframe for TDD (SSUBframe) , all other subframes (NSUBframe) .

param cell_name
No help available

param mcs
Comma-separated list of 16 MCS values, for reported CQI values 0 to 15.

6.3.4.7.2.174 McsTable

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:CMMapping:SSUBframe:MCSTable
```

class McsTableCls

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

class GetStruct

Response structure. Fields:

- **Mcs_Table: enums.McsTableC:**
  - AUTO: Mapping table selected automatically, depending on [CONFigure:]SIGNaling:LTE:CELL:UEScheduling:UDEFined:SASSignment:DL:MCSTable.
  - P521: Mapping table contents defined by 3GPP TS 36.521.MCS scheme selection via Predefined3GPP.
  - UDEFined: Mapping table contents defined via separate command, see [CONFigure:]SIGNaling:LTE:CELL:UEScheduling:CMMapping:CSIRs:MCS etc.
- Predefined_3_Gpp: enums.Predefined3Gpp: Selects an MCS scheme for MCSTable = P521. Mn means 'MCS.n' in the tables A.

4-13 to A.4-16 in 3GPP TS 36.521.

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:CMMapping:SSUBframe:MCSTable
value: GetStruct = driver.configure.signaling.lte.cell.ueScheduling.cmMapping.
↪ssubframe.mcsTable.get(cell_name = 'abc')
```

Selects a configuration mode for the CQI-MCS mapping tables for follow WB CQI. There is a mapping table for each type of DL subframe: CSI-RS subframe (CSIRs) , special subframe for TDD (SSUBframe) , all other subframes (NSUBframe) .

param cell_name
No help available

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

set(*cell_name: str*, *mcs_table: McsTableC*, *predefined_3_gpp: Predefined3Gpp = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:CMMapping:SSUBframe:MCSTable
driver.configure.signaling.lte.cell.ueScheduling.cmMapping.ssubframe.mcsTable.
↪set(cell_name = 'abc', mcs_table = enums.McsTableC.AUTO, predefined_3_gpp =␣
↪enums.Predefined3Gpp.M1)
```

Selects a configuration mode for the CQI-MCS mapping tables for follow WB CQI. There is a mapping table for each type of DL subframe: CSI-RS subframe (CSIRs) , special subframe for TDD (SSUBframe) , all other subframes (NSUBframe) .

param cell_name
No help available

param mcs_table

- AUTO: Mapping table selected automatically, depending on [CONFigure:]SIGNaling:LTE:CELL:UEScheduling:UDEFined:SASSignment:DL:MCSTable.
- P521: Mapping table contents defined by 3GPP TS 36.521.MCS scheme selection via Predefined3GPP.
- UDEFined: Mapping table contents defined via separate command, see [CONFigure:]SIGNaling:LTE:CELL:UEScheduling:CMMapping:CSIRs:MCS etc.

param predefined_3_gpp
Selects an MCS scheme for MCSTable = P521. Mn means 'MCS.n' in the tables A.4-13 to A.4-16 in

3GPP TS 36.521.

6.3.4.7.2.175 Downlink

class DownlinkCls

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

Cloning the Group

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

Subgroups

6.3.4.7.2.176 Smode

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:DL:SMODe
```

class SmodeCls

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

get(*cell_name: str*) → ModeE

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:DL:SMODe
value: enums.ModeE = driver.configure.signaling.lte.cell.ueScheduling.downlink.
↪smode.get(cell_name = 'abc')
```

Selects a scheduling mode for the DL.

param cell_name
No help available

return
mode: FIXed: Fixed scheduling SPS: Semi-persistent scheduling CQI: Follow CQI WB PMI: Follow PMI WB CRI: Follow CQI WB + RI RI: Follow RI CPRI: Follow CQI WB + PMI WB + RI PRI: Follow PMI WB + RI UDEFined: Other scheduling mode (query only) .

set(*cell_name: str*, *mode: ModeE*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:DL:SMODe
driver.configure.signaling.lte.cell.ueScheduling.downlink.smode.set(cell_name =
↪'abc', mode = enums.ModeE.CPRI)
```

Selects a scheduling mode for the DL.

param cell_name
No help available

param mode
FIXed: Fixed scheduling SPS: Semi-persistent scheduling CQI: Follow CQI WB PMI: Follow PMI WB CRI: Follow CQI WB + RI RI: Follow RI CPRI: Follow CQI WB + PMI WB + RI PRI: Follow PMI WB + RI UDEFined: Other scheduling mode (query only) .

6.3.4.7.2.177 Laa

class LaaCls

Laa commands group definition. 28 total commands, 10 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.4.7.2.178 Crate

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:LAA:CRATe
```

class CrateCls

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

get(*cell_name: str*, *subframe: int*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:LAA:CRATe
value: int = driver.configure.signaling.lte.cell.ueScheduling.laa.crate.
↪get(cell_name = 'abc', subframe = 1)
```

Queries the code rate for LAA subframes with <Subframe> allocated symbols.

param cell_name
No help available

param subframe
No help available

return
modulation: No help available

6.3.4.7.2.179 Csat

class CsatCls

Csat commands group definition. 4 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.lte.cell.ueScheduling.laa.csat.clone()
```

Subgroups

6.3.4.7.2.180 Dmtc

class DmtcCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.lte.cell.ueScheduling.laa.csat.dmtc.clone()
```

Subgroups

6.3.4.7.2.181 Period

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:CSAT:DMTC:PERiod
```

class PeriodCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:CSAT:DMTC:PERiod
value: int = driver.configure.signaling.lte.cell.ueScheduling.laa.csat.dmtc.
↪period.get(cell_name = 'abc')
```

Configures the periodicity for transmission of a discovery signal to the UE, for LAA.

param cell_name
No help available

return
periodicity: No help available

set(*cell_name: str*, *periodicity: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:CSAT:DMTC:PERiod
driver.configure.signaling.lte.cell.ueScheduling.laa.csat.dmtc.period.set(cell_
↪name = 'abc', periodicity = 1)
```

Configures the periodicity for transmission of a discovery signal to the UE, for LAA.

param cell_name
No help available

param periodicity
No help available

6.3.4.7.2.182 Soffset

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:LAA:CSAT:DMTC:SOFFset
```

class SoffsetCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:LAA:CSAT:DMTC:SOFFset
value: int = driver.configure.signaling.lte.cell.ueScheduling.laa.csat.dmtc.
↪soffset.get(cell_name = 'abc')
```

Configures the start offset for transmission of a discovery signal to the UE, for LAA.

param cell_name
No help available

return
offset: No help available

set(*cell_name: str*, *offset: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:LAA:CSAT:DMTC:SOFFset
driver.configure.signaling.lte.cell.ueScheduling.laa.csat.dmtc.soffset.set(cell_
↪name = 'abc', offset = 1)
```

Configures the start offset for transmission of a discovery signal to the UE, for LAA.

param cell_name
No help available

param offset
No help available

6.3.4.7.2.183 DsOccasion

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:LAA:CSAT:DSOCcasion
```

class DsOccasionCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:LAA:CSAT:DSOCcasion
value: int = driver.configure.signaling.lte.cell.ueScheduling.laa.csat.
↪dsOccasion.get(cell_name = 'abc')
```

Configures the duration of discovery signal occasions, for LAA.

param cell_name
No help available

return
occasion: No help available

set(*cell_name: str*, *occasion: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:CSAT:DSOCcasion
driver.configure.signaling.lte.cell.ueScheduling.laa.csat.dsOccasion.set(cell_
↪name = 'abc', occasion = 1)
```

Configures the duration of discovery signal occasions, for LAA.

param cell_name
No help available

param occasion
No help available

6.3.4.7.2.184 Enable

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:CSAT:ENABle
```

class EnableCls

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

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:CSAT:ENABle
value: bool = driver.configure.signaling.lte.cell.ueScheduling.laa.csat.enable.
↪get(cell_name = 'abc')
```

Enables CSAT and SCell muting for LAA.

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:CSAT:ENABle
driver.configure.signaling.lte.cell.ueScheduling.laa.csat.enable.set(cell_name␣
↪= 'abc', enable = False)
```

Enables CSAT and SCell muting for LAA.

param cell_name
No help available

param enable
No help available

6.3.4.7.2.185 Cword<Cword>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.configure.signaling.lte.cell.ueScheduling.laa.cword.repcap_cword_get()
driver.configure.signaling.lte.cell.ueScheduling.laa.cword.repcap_cword_set(repcap.Cword.
↪Nr1)
```

class CwordCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.lte.cell.ueScheduling.laa.cword.clone()
```

Subgroups

6.3.4.7.2.186 Mcs

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:CWORd<no>:MCS
```

class McsCls

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

get(*cell_name: str*, *subframe: int*, *cword=Cword.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:CWORd<no>:MCS
value: int = driver.configure.signaling.lte.cell.ueScheduling.laa.cword.mcs.
↪get(cell_name = 'abc', subframe = 1, cword = repcap.Cword.Default)
```

Configures the MCS index for LAA subframes with <Subframe> allocated symbols, code word <no>.

param cell_name
No help available

param subframe
No help available

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

return
mcs: No help available

set(*cell_name: str*, *subframe: int*, *mcs: int*, *cword=Cword.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:CWORd<no>:MCS
driver.configure.signaling.lte.cell.ueScheduling.laa.cword.mcs.set(cell_name =
↪'abc', subframe = 1, mcs = 1, cword = repcap.Cword.Default)
```

Configures the MCS index for LAA subframes with <Subframe> allocated symbols, code word <no>.

param cell_name
No help available

param subframe
No help available

param mcs
No help available

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

6.3.4.7.2.187 Modulation

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:CWORd<no>:MODulation
```

class ModulationCls

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

get(*cell_name: str*, *subframe: int*, *cword=Cword.Default*) → Modulation

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:CWORd<no>:MODulation
value: enums.Modulation = driver.configure.signaling.lte.cell.ueScheduling.laa.
↪cword.modulation.get(cell_name = 'abc', subframe = 1, cword = repcap.Cword.
↪Default)
```

Queries the modulation scheme for LAA subframes with <Subframe> allocated symbols, code word <no>.

param cell_name
No help available

param subframe
No help available

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

return
modulation: No help available

6.3.4.7.2.188 DciFormat

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:DCIFormat
```

class DciFormatCls

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

get(*cell_name: str*, *subframe: int*) → DciFormat

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:DCIFormat
value: enums.DciFormat = driver.configure.signaling.lte.cell.ueScheduling.laa.
↪dciFormat.get(cell_name = 'abc', subframe = 1)
```

Configures the DCI format for LAA subframes with <Subframe> allocated symbols.

param cell_name
No help available

param subframe
No help available

return
dci_format: No help available

set(*cell_name: str*, *subframe: int*, *dci_format: DciFormat*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:DCIFormat
driver.configure.signaling.lte.cell.ueScheduling.laa.dciFormat.set(cell_name =
↪'abc', subframe = 1, dci_format = enums.DciFormat.D0)
```

Configures the DCI format for LAA subframes with <Subframe> allocated symbols.

param cell_name
No help available

param subframe
No help available

param dci_format
No help available

6.3.4.7.2.189 Fburst

class FburstCls

Fburst commands group definition. 8 total commands, 7 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.lte.cell.ueScheduling.laa.fburst.clone()
```

Subgroups

6.3.4.7.2.190 All

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:LAA:FBURst:ALL
```

class AllCls

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

class GetStruct

Response structure. Fields:

- Periodicity: int: No parameter help available
- Burst_Length: int: No parameter help available
- First_Subframe: int: No parameter help available
- Initial_Sf_Alloc: enums.InitialSfAlloc: Initial subframe with full allocation (S0) or second slot only (S7)
- Ofdm_Symbols: enums.OfdmSymbols: OFDM symbols in last subframe of burst
- Ccrntis_End: enums.CcrntisEnd: Send CC-RNTI: F2SF: final 2 SF LSF: last SF BLSF: before last SF SASF: skip all SF ASF: all SF
- Pdcch_Format: enums.PdcchFormatB: Number of CCE for PDCCH scrambled with CC-RNTI.

class SetStruct

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

- Cell_Name: str: No parameter help available
- Periodicity: int: No parameter help available
- Burst_Length: int: No parameter help available
- First_Subframe: int: No parameter help available
- Initial_Sf_Alloc: enums.InitialSfAlloc: Optional setting parameter. Initial subframe with full allocation (S0) or second slot only (S7)
- Ofdm_Symbols: enums.OfdmSymbols: Optional setting parameter. OFDM symbols in last subframe of burst
- Ccrntis_End: enums.CcrntisEnd: Optional setting parameter. Send CC-RNTI: F2SF: final 2 SF LSF: last SF BLSF: before last SF SASF: skip all SF ASF: all SF
- Pdcch_Format: enums.PdcchFormatB: Optional setting parameter. Number of CCE for PDCCH scrambled with CC-RNTI.

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:FBURst:ALL
value: GetStruct = driver.configure.signaling.lte.cell.ueScheduling.laa.fburst.
↪all.get(cell_name = 'abc')
```

Configures fixed bursts (combination of the other ...:FBURst:... commands) .

param cell_name
No help available

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

set(*structure: SetStruct*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:FBURst:ALL
structure = driver.configure.signaling.lte.cell.ueScheduling.laa.fburst.all.
↪SetStruct()
structure.Cell_Name: str = 'abc'
structure.Periodicity: int = 1
structure.Burst_Length: int = 1
structure.First_Subframe: int = 1
structure.Initial_Sf_Alloc: enums.InitialSfAlloc = enums.InitialSfAlloc.S0
structure.Ofdm_Symbols: enums.OfdmSymbols = enums.OfdmSymbols.ALL
structure.Ccrntis_End: enums.CcrntisEnd = enums.CcrntisEnd.ASF
structure.Pdcch_Format: enums.PdcchFormatB = enums.PdcchFormatB.N1
driver.configure.signaling.lte.cell.ueScheduling.laa.fburst.all.set(structure)
```

Configures fixed bursts (combination of the other ...:FBURst:... commands) .

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

6.3.4.7.2.191 Blength

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:FBURst:BLENgth
```

class BlengthCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:FBURst:BLENgth
value: int = driver.configure.signaling.lte.cell.ueScheduling.laa.fburst.
↪blength.get(cell_name = 'abc')
```

Configures the length of fixed bursts (number of subframes) .

param cell_name
No help available

return
burst_length: No help available

set(*cell_name: str*, *burst_length: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:FBURst:BLENgth
driver.configure.signaling.lte.cell.ueScheduling.laa.fburst.blength.set(cell_
↪name = 'abc', burst_length = 1)
```

Configures the length of fixed bursts (number of subframes) .

param cell_name
No help available

param burst_length
No help available

6.3.4.7.2.192 Ccrnti

class CcrntiCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.lte.cell.ueScheduling.laa.fburst.ccrnti.clone()
```

Subgroups

6.3.4.7.2.193 PdcchFormat

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:FBURst:CCRNti:PDCChformat
```

class PdcchFormatCls

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

get(*cell_name: str*) → PdcchFormatB

```
# SCPI:␣
↪[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:FBURst:CCRNti:PDCChformat
value: enums.PdcchFormatB = driver.configure.signaling.lte.cell.ueScheduling.
↪laa.fburst.ccrnti.pdcchFormat.get(cell_name = 'abc')
```

Selects the number of control channel elements (CCE) used for transmission of the PDCCH scrambled with CC-RNTI, for fixed bursts.

param cell_name
No help available

return
pdcch_format: No help available

set(*cell_name: str*, *pdcch_format: PdcchFormatB*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:FBURst:CCRNti:PDCChformat
driver.configure.signaling.lte.cell.ueScheduling.laa.fburst.ccrnti.pdcchFormat.
↪set(cell_name = 'abc', pdcch_format = enums.PdcchFormatB.N1)
```

Selects the number of control channel elements (CCE) used for transmission of the PDCCH scrambled with CC-RNTI, for fixed bursts.

param cell_name
No help available

param pdcch_format
No help available

6.3.4.7.2.194 Send

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:FBURst:CCRNti:SEND
```

class SendCls

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

get(*cell_name: str*) → CcrntisEnd

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:FBURst:CCRNti:SEND
value: enums.CcrntisEnd = driver.configure.signaling.lte.cell.ueScheduling.laa.
↪fburst.ccrnti.send.get(cell_name = 'abc')
```

Selects subframes for transmission of CC-RNTI, for fixed bursts.

param cell_name
No help available

return
ccrntis_end: Send CC-RNTI: F2SF: final 2 SF LSF: last SF BLSF: before last SF SASF: skip all SF ASF: all SF

set(*cell_name: str*, *ccrntis_end: CcrntisEnd*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:FBURst:CCRNti:SEND
driver.configure.signaling.lte.cell.ueScheduling.laa.fburst.ccrnti.send.
↪set(cell_name = 'abc', ccrntis_end = enums.CcrntisEnd.ASF)
```

Selects subframes for transmission of CC-RNTI, for fixed bursts.

param cell_name
No help available

param ccrntis_end
Send CC-RNTI: F2SF: final 2 SF LSF: last SF BLSF: before last SF SASF: skip all SF ASF: all SF

6.3.4.7.2.195 FsBurst

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:LAA:FBURst:FSBurst
```

class FsBurstCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:LAA:FBURst:FSBurst
value: int = driver.configure.signaling.lte.cell.ueScheduling.laa.fburst.
↪fsBurst.get(cell_name = 'abc')
```

Selects the first subframe used for fixed bursts. Zero refers to subframe 0 in system frame 0 (SFN 0) .

param cell_name
No help available

return
first_subframe: No help available

set(*cell_name: str*, *first_subframe: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:LAA:FBURst:FSBurst
driver.configure.signaling.lte.cell.ueScheduling.laa.fburst.fsBurst.set(cell_
↪name = 'abc', first_subframe = 1)
```

Selects the first subframe used for fixed bursts. Zero refers to subframe 0 in system frame 0 (SFN 0) .

param cell_name
No help available

param first_subframe
No help available

6.3.4.7.2.196 IsaBurst

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:LAA:FBURst:ISABurst
```

class IsaBurstCls

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

get(*cell_name: str*) → InitialSfAlloc

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:LAA:FBURst:ISABurst
value: enums.InitialSfAlloc = driver.configure.signaling.lte.cell.ueScheduling.
↪laa.fburst.isaBurst.get(cell_name = 'abc')
```

Selects the first allocated symbol for the first subframe of a fixed burst.

param cell_name
No help available

return
initial_sf_alloc: Symbol 0 (full allocation of subframe) or symbol 7 (second slot of subframe)

set(*cell_name: str*, *initial_sf_alloc: InitialSfAlloc*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:FBURst:ISABurst
driver.configure.signaling.lte.cell.ueScheduling.laa.fburst.isaBurst.set(cell_
↪name = 'abc', initial_sf_alloc = enums.InitialSfAlloc.S0)
```

Selects the first allocated symbol for the first subframe of a fixed burst.

param cell_name
No help available

param initial_sf_alloc
Symbol 0 (full allocation of subframe) or symbol 7 (second slot of subframe)

6.3.4.7.2.197 OslSubframe

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:FBURst:OSLSubframe
```

class OslSubframeCls

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

get(*cell_name: str*) → OfdmSymbols

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:FBURst:OSLSubframe
value: enums.OfdmSymbols = driver.configure.signaling.lte.cell.ueScheduling.laa.
↪fburst.oslSubframe.get(cell_name = 'abc')
```

Selects the number of allocated OFDM symbols at the beginning of the last subframe of a fixed burst.

param cell_name
No help available

return
ofdm_symbols: No help available

set(*cell_name: str*, *ofdm_symbols: OfdmSymbols*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:FBURst:OSLSubframe
driver.configure.signaling.lte.cell.ueScheduling.laa.fburst.oslSubframe.
↪set(cell_name = 'abc', ofdm_symbols = enums.OfdmSymbols.ALL)
```

Selects the number of allocated OFDM symbols at the beginning of the last subframe of a fixed burst.

param cell_name
No help available

param ofdm_symbols
No help available

6.3.4.7.2.198 Pbtr

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:LAA:FBURst:PBTR
```

class PbtrCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:LAA:FBURst:PBTR
value: int = driver.configure.signaling.lte.cell.ueScheduling.laa.fburst.pbtr.
↪get(cell_name = 'abc')
```

Configures the periodicity n for subsequent fixed LAA bursts (a burst starts every nth subframe) .

param cell_name
No help available

return
periodicity: No help available

set(*cell_name: str*, *periodicity: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:LAA:FBURst:PBTR
driver.configure.signaling.lte.cell.ueScheduling.laa.fburst.pbtr.set(cell_name␣
↪= 'abc', periodicity = 1)
```

Configures the periodicity n for subsequent fixed LAA bursts (a burst starts every nth subframe) .

param cell_name
No help available

param periodicity
No help available

6.3.4.7.2.199 PdcchFormat

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:LAA:PDCChformat
```

class PdcchFormatCls

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

get(*cell_name: str*, *subframe: int*) → PdcchFormat

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:LAA:PDCChformat
value: enums.PdcchFormat = driver.configure.signaling.lte.cell.ueScheduling.laa.
↪pdcchFormat.get(cell_name = 'abc', subframe = 1)
```

Queries the number of control channel elements (CCE) used for transmission of the PDCCH, for LAA subframes with <Subframe> allocated symbols.

param cell_name
No help available

param subframe
No help available

return
pdcch_format: No help available

6.3.4.7.2.200 Rb

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:LAA:RB
```

class RbCls

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

class GetStruct

Response structure. Fields:

- Number_Rb: int: No parameter help available
- Start_Rb: int: No parameter help available

get(*cell_name: str*, *subframe: int*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:LAA:RB
value: GetStruct = driver.configure.signaling.lte.cell.ueScheduling.laa.rb.
↪get(cell_name = 'abc', subframe = 1)
```

Configures the RB allocation for LAA subframes with <Subframe> allocated symbols.

param cell_name
No help available

param subframe
No help available

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

set(*cell_name: str*, *subframe: int*, *number_rb: int*, *start_rb: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:LAA:RB
driver.configure.signaling.lte.cell.ueScheduling.laa.rb.set(cell_name = 'abc',␣
↪subframe = 1, number_rb = 1, start_rb = 1)
```

Configures the RB allocation for LAA subframes with <Subframe> allocated symbols.

param cell_name
No help available

param subframe
No help available

param number_rb
No help available

param start_rb
No help available

6.3.4.7.2.201 Rburst

class RburstCls

Rburst commands group definition. 8 total commands, 7 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.lte.cell.ueScheduling.laa.rburst.clone()
```

Subgroups

6.3.4.7.2.202 All

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:LAA:RBURst:ALL
```

class AllCls

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

class GetStruct

Response structure. Fields:

- Periodicity: int: No parameter help available
- Bt_Ratio: float: Burst transmission probability
- Ips_Ratio: float: Initial partial subframes probability
- Ccrntis_End: enums.CcrntisEnd: Send CC-RNTI: F2SF: final 2 SF LSF: last SF BLSF: before last SF SASF: skip all SF ASF: all SF
- Pdcch_Format: enums.PdcchFormatB: Number of CCE for PDCCH scrambled with CC-RNTI.
- Bl_Count: int: Number of values for BurstLength
- Scount: int: Number of values for Symbols
- Burst_Length: List[int]: Comma-separated list of all allowed values for the number of subframes in a random burst
- Symbols: List[int]: Comma-separated list of all allowed values for the number of allocated OFDM symbols in the last subframe

class SetStruct

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

- Cell_Name: str: No parameter help available
- Periodicity: int: No parameter help available

- Ccrntis_End: enums.CcrntisEnd: Optional setting parameter. Send CC-RNTI: F2SF: final 2 SF LSF: last SF BLSF: before last SF SASF: skip all SF ASF: all SF
- Pdcch_Format: enums.PdcchFormatB: Optional setting parameter. Number of CCE for PDCCH scrambled with CC-RNTI.
- Bl_Count: int: Optional setting parameter. Number of values for BurstLength
- Scount: int: Optional setting parameter. Number of values for Symbols
- Burst_Length: List[int]: Optional setting parameter. Comma-separated list of all allowed values for the number of subframes in a random burst
- Symbols: List[int]: Optional setting parameter. Comma-separated list of all allowed values for the number of allocated OFDM symbols in the last subframe

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:RBURst:ALL
value: GetStruct = driver.configure.signaling.lte.cell.ueScheduling.laa.rburst.
↪all.get(cell_name = 'abc')
```

This command combines the other . . . :RBURst:. . . commands to configure random bursts. A query returns: <Periodicity>, <BTRatio>, <IPSRatio>, <CCRNTISend>, <PDCCHFormat>, <BLCount>, <SCount>, <BurstLength>, <Symbols>

param cell_name
No help available

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

set(*structure: SetStruct*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:RBURst:ALL
structure = driver.configure.signaling.lte.cell.ueScheduling.laa.rburst.all.
↪SetStruct()
structure.Cell_Name: str = 'abc'
structure.Periodicity: int = 1
structure.Ccrntis_End: enums.CcrntisEnd = enums.CcrntisEnd.ASF
structure.Pdcch_Format: enums.PdcchFormatB = enums.PdcchFormatB.N1
structure.Bl_Count: int = 1
structure.Scount: int = 1
structure.Burst_Length: List[int] = [1, 2, 3]
structure.Symbols: List[int] = [1, 2, 3]
driver.configure.signaling.lte.cell.ueScheduling.laa.rburst.all.set(structure)
```

This command combines the other . . . :RBURst:. . . commands to configure random bursts. A query returns: <Periodicity>, <BTRatio>, <IPSRatio>, <CCRNTISend>, <PDCCHFormat>, <BLCount>, <SCount>, <BurstLength>, <Symbols>

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

6.3.4.7.2.203 Blength

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:LAA:RBURst:BLENgth
```

class BlengthCls

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

get(*cell_name: str*) → List[int]

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:LAA:RBURst:BLENgth
value: List[int] = driver.configure.signaling.lte.cell.ueScheduling.laa.rburst.
↪blength.get(cell_name = 'abc')
```

Selects allowed values for the number of subframes in a random burst.

param cell_name
No help available

return
burst_length: Comma-separated list of all allowed values

set(*cell_name: str*, *burst_length: List[int]*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:LAA:RBURst:BLENgth
driver.configure.signaling.lte.cell.ueScheduling.laa.rburst.blength.set(cell_
↪name = 'abc', burst_length = [1, 2, 3])
```

Selects allowed values for the number of subframes in a random burst.

param cell_name
No help available

param burst_length
Comma-separated list of all allowed values

6.3.4.7.2.204 BtProb

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:LAA:RBURst:BTPRob
```

class BtProbCls

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

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:LAA:RBURst:BTPRob
value: float = driver.configure.signaling.lte.cell.ueScheduling.laa.rburst.
↪btProb.get(cell_name = 'abc')
```

Queries the probability for the decision of the random transmission procedure, whether a random burst is transmitted or whether muting is applied for the burst duration.

param cell_name
No help available

return
ratio: No help available

6.3.4.7.2.205 Ccrnti

class CcrntiCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.lte.cell.ueScheduling.laa.rburst.ccrnti.clone()
```

Subgroups

6.3.4.7.2.206 PdcchFormat

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:RBURst:CCRNti:PDCChformat
```

class PdcchFormatCls

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

get(*cell_name: str*) → PdcchFormatB

```
# SCPI:␣
↪[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:RBURst:CCRNti:PDCChformat
value: enums.PdcchFormatB = driver.configure.signaling.lte.cell.ueScheduling.
↪laa.rburst.ccrnti.pdcchFormat.get(cell_name = 'abc')
```

Selects the number of control channel elements (CCE) used for transmission of the PDCCH scrambled with CC-RNTI, for random bursts.

param cell_name
No help available

return
pdcch_format: No help available

set(*cell_name: str*, *pdcch_format: PdcchFormatB*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:RBURst:CCRNti:PDCChformat
driver.configure.signaling.lte.cell.ueScheduling.laa.rburst.ccrnti.pdcchFormat.
↪set(cell_name = 'abc', pdcch_format = enums.PdcchFormatB.N1)
```

Selects the number of control channel elements (CCE) used for transmission of the PDCCH scrambled with CC-RNTI, for random bursts.

param cell_name
No help available

param pdcch_format
No help available

6.3.4.7.2.207 Send

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:LAA:RBURst:CCRNti:SEND
```

class SendCls

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

get(*cell_name: str*) → CcrntisEnd

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:LAA:RBURst:CCRNti:SEND
value: enums.CcrntisEnd = driver.configure.signaling.lte.cell.ueScheduling.laa.
↪rburst.ccrnti.send.get(cell_name = 'abc')
```

Selects subframes for transmission of CC-RNTI, for random bursts.

param cell_name
No help available

return
ccrntis_end: Send CC-RNTI: F2SF: final 2 SF LSF: last SF BLSF: before last SF SASF: skip all SF ASF: all SF

set(*cell_name: str*, *ccrntis_end: CcrntisEnd*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:LAA:RBURst:CCRNti:SEND
driver.configure.signaling.lte.cell.ueScheduling.laa.rburst.ccrnti.send.
↪set(cell_name = 'abc', ccrntis_end = enums.CcrntisEnd.ASF)
```

Selects subframes for transmission of CC-RNTI, for random bursts.

param cell_name
No help available

param ccrntis_end
Send CC-RNTI: F2SF: final 2 SF LSF: last SF BLSF: before last SF SASF: skip all SF ASF: all SF

6.3.4.7.2.208 IpsProb

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:LAA:RBURst:IPSProb
```

class IpsProbCls

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

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:RBURst:IPSProb
value: float = driver.configure.signaling.lte.cell.ueScheduling.laa.rburst.
↪ipsProb.get(cell_name = 'abc')
```

Queries the probability for the decision of the random transmission procedure, whether the first subframe of a random burst has a partial allocation (instead of a full allocation) .

param cell_name
No help available

return
ratio: No help available

6.3.4.7.2.209 Pbtr

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:RBURst:PBTR
```

class PbtrCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:RBURst:PBTR
value: int = driver.configure.signaling.lte.cell.ueScheduling.laa.rburst.pbtr.
↪get(cell_name = 'abc')
```

Configures the periodicity n for subsequent random LAA bursts (a burst starts every nth subframe) .

param cell_name
No help available

return
periodicity: No help available

set(*cell_name: str*, *periodicity: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:RBURst:PBTR
driver.configure.signaling.lte.cell.ueScheduling.laa.rburst.pbtr.set(cell_name␣
↪= 'abc', periodicity = 1)
```

Configures the periodicity n for subsequent random LAA bursts (a burst starts every nth subframe) .

param cell_name
No help available

param periodicity
No help available

6.3.4.7.2.210 PlSubframe

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:LAA:RBURst:PLSubframe
```

class PlSubframeCls

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

get(*cell_name: str*) → List[int]

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:LAA:RBURst:PLSubframe
value: List[int] = driver.configure.signaling.lte.cell.ueScheduling.laa.rburst.
↪plSubframe.get(cell_name = 'abc')
```

Selects allowed values for the number of allocated OFDM symbols in the last subframe of a random burst.

param cell_name
No help available

return
symbols: Comma-separated list of all allowed values

set(*cell_name: str*, *symbols: List[int]*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:LAA:RBURst:PLSubframe
driver.configure.signaling.lte.cell.ueScheduling.laa.rburst.plSubframe.set(cell_
↪name = 'abc', symbols = [1, 2, 3])
```

Selects allowed values for the number of allocated OFDM symbols in the last subframe of a random burst.

param cell_name
No help available

param symbols
Comma-separated list of all allowed values

6.3.4.7.2.211 Riv

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:LAA:RIV
```

class RivCls

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

get(*cell_name: str*, *subframe: int*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:LAA:RIV
value: int = driver.configure.signaling.lte.cell.ueScheduling.laa.riv.get(cell_
↪name = 'abc', subframe = 1)
```

Configures the resource indication value (RIV) for LAA subframes with <Subframe> allocated symbols.

param cell_name
No help available

param subframe
No help available

return
riv: No help available

set(*cell_name: str*, *subframe: int*, *riv: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:RIV
driver.configure.signaling.lte.cell.ueScheduling.laa.riv.set(cell_name = 'abc',␣
↪subframe = 1, riv = 1)
```

Configures the resource indication value (RIV) for LAA subframes with <Subframe> allocated symbols.

param cell_name
No help available

param subframe
No help available

param riv
No help available

6.3.4.7.2.212 Tbursts

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:TBURsts
```

class TburstsCls

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

get(*cell_name: str*) → BurstType

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:TBURsts
value: enums.BurstType = driver.configure.signaling.lte.cell.ueScheduling.laa.
↪tbursts.get(cell_name = 'abc')
```

Selects the burst type for LAA.

param cell_name
No help available

return
burst_type: Fixed bursts or random bursts

set(*cell_name: str*, *burst_type: BurstType*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:LAA:TBURsts
driver.configure.signaling.lte.cell.ueScheduling.laa.tbursts.set(cell_name =
↪'abc', burst_type = enums.BurstType.FBURst)
```

Selects the burst type for LAA.

param cell_name
No help available

param burst_type
Fixed bursts or random bursts

6.3.4.7.2.213 Rmc

class RmcCls

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

Cloning the Group

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

Subgroups

6.3.4.7.2.214 Downlink

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:RMC:DL
```

class DownlinkCls

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

class GetStruct

Response structure. Fields:

- Enable: bool: Enables or disables scheduling for all DL subframes.
- Modulation: enums.ModulationB: No parameter help available
- Number_Rb: int: No parameter help available
- Start_Rb: int: No parameter help available

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:RMC:DL
value: GetStruct = driver.configure.signaling.lte.cell.ueScheduling.rmc.
↪downlink.get(cell_name = 'abc')
```

Configures LTE cell settings to values compliant with a DL RMC definition. A setting command accepts only certain value combinations. Use the RMC wizard in the GUI to get allowed value combinations. A query returns the set of values that are presented by the RMC wizard. These values can differ from currently applied values. Omit optional parameters only if you do not care which value you get (just any RMC-compliant value) .

param cell_name
No help available

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

set(*cell_name: str*, *enable: bool*, *modulation: ModulationB = None*, *number_rb: int = None*, *start_rb: int = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:RMC:DL
driver.configure.signaling.lte.cell.ueScheduling.rmc.downlink.set(cell_name =
↪'abc', enable = False, modulation = enums.ModulationB.BPSK, number_rb = 1,␣
↪start_rb = 1)
```

Configures LTE cell settings to values compliant with a DL RMC definition. A setting command accepts only certain value combinations. Use the RMC wizard in the GUI to get allowed value combinations. A query returns the set of values that are presented by the RMC wizard. These values can differ from currently applied values. Omit optional parameters only if you do not care which value you get (just any RMC-compliant value) .

param cell_name
No help available

param enable
Enables or disables scheduling for all DL subframes.

param modulation
No help available

param number_rb
No help available

param start_rb
No help available

6.3.4.7.2.215 Uplink

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:RMC:UL
```

class UplinkCls

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

class GetStruct

Response structure. Fields:

- Enable: bool: Enables scheduling for all UL subframes.
- Modulation: enums.ModulationB: No parameter help available
- Number_Rb: int: No parameter help available
- Start_Rb: int: No parameter help available

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:RMC:UL
value: GetStruct = driver.configure.signaling.lte.cell.ueScheduling.rmc.uplink.
↪get(cell_name = 'abc')
```

Configures LTE cell settings to values compliant with a UL RMC definition. A setting command accepts only certain value combinations. Use the RMC wizard in the GUI to get allowed value combinations.

A query returns the set of values that are presented by the RMC wizard. These values can differ from currently applied values. Omit optional parameters only if you do not care which value you get (just any RMC-compliant value) .

param cell_name
No help available

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

set(*cell_name: str*, *enable: bool*, *modulation: ModulationB = None*, *number_rb: int = None*, *start_rb: int = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:RMC:UL
driver.configure.signaling.lte.cell.ueScheduling.rmc.uplink.set(cell_name = 'abc
↪', enable = False, modulation = enums.ModulationB.BPSK, number_rb = 1, start_
↪rb = 1)
```

Configures LTE cell settings to values compliant with a UL RMC definition. A setting command accepts only certain value combinations. Use the RMC wizard in the GUI to get allowed value combinations. A query returns the set of values that are presented by the RMC wizard. These values can differ from currently applied values. Omit optional parameters only if you do not care which value you get (just any RMC-compliant value) .

param cell_name
No help available

param enable
Enables scheduling for all UL subframes.

param modulation
No help available

param number_rb
No help available

param start_rb
No help available

6.3.4.7.2.216 Smode

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:SMODe
```

class SmodeCls

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

get(*cell_name: str*) → ModeE

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:SMODe
value: enums.ModeE = driver.configure.signaling.lte.cell.ueScheduling.smode.
↪get(cell_name = 'abc')
```

No command help available

param cell_name
No help available

return
mode: No help available

set(*cell_name: str*, *mode: ModeE*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:SMODe
driver.configure.signaling.lte.cell.ueScheduling.smode.set(cell_name = 'abc',␣
↪mode = enums.ModeE.CPRI)
```

No command help available

param cell_name
No help available

param mode
No help available

6.3.4.7.2.217 Sps

class SpsCls

Sps commands group definition. 13 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.4.7.2.218 Sassignment

class SassignmentCls

Sassignment commands group definition. 13 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.lte.cell.ueScheduling.sps.sassignment.clone()
```

Subgroups

6.3.4.7.2.219 Downlink

class DownlinkCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.lte.cell.ueScheduling.sps.sassignment.downlink.
↪clone()
```

Subgroups

6.3.4.7.2.220 All

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:SPS:SASSignment:DL:ALL
```

class AllCls

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

class GetStruct

Response structure. Fields:

- Sf_Interval: int: Subframe periodicity
- Number_Rb: int: No parameter help available
- Start_Rb: int: No parameter help available
- Mcs: int: No parameter help available
- Tbs_Bits: int: No parameter help available

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:SPS:SASSignment:DL:ALL
value: GetStruct = driver.configure.signaling.lte.cell.ueScheduling.sps.
↪sassignment.downlink.all.get(cell_name = 'abc')
```

Configures all settings for SPS DL scheduling. A query returns the sequence <SFInterval>, <NumberRB>, <StartRB>, <MCS>, <TBSBits>.

param cell_name
No help available

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

set(*cell_name: str*, *sf_interval: int = None*, *number_rb: int = None*, *start_rb: int = None*, *mcs: int = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:SPS:SASSignment:DL:ALL
driver.configure.signaling.lte.cell.ueScheduling.sps.sassignment.downlink.all.
↪set(cell_name = 'abc', sf_interval = 1, number_rb = 1, start_rb = 1, mcs = 1)
```

Configures all settings for SPS DL scheduling. A query returns the sequence <SFInterval>, <NumberRB>, <StartRB>, <MCS>, <TBSBits>.

param cell_name
No help available

param sf_interval
Subframe periodicity

param number_rb
No help available

param start_rb
No help available

param mcs
No help available

6.3.4.7.2.221 Mcs

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:SPS:SASSignment:DL:MCS
```

class McsCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:SPS:SASSignment:DL:MCS
value: int = driver.configure.signaling.lte.cell.ueScheduling.sps.sassignment.
↪downlink.mcs.get(cell_name = 'abc')
```

Specifies the MCS index for SPS DL scheduling.

param cell_name
No help available

return
mcs: No help available

set(*cell_name: str*, *mcs: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:SPS:SASSignment:DL:MCS
driver.configure.signaling.lte.cell.ueScheduling.sps.sassignment.downlink.mcs.
↪set(cell_name = 'abc', mcs = 1)
```

Specifies the MCS index for SPS DL scheduling.

param cell_name
No help available

param mcs
No help available

6.3.4.7.2.222 Rb

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:SPS:SASSignment:DL:RB
```

class RbCls

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

class GetStruct

Response structure. Fields:

- Number_Rb: int: No parameter help available
- Start_Rb: int: No parameter help available

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:SPS:SASSignment:DL:RB
value: GetStruct = driver.configure.signaling.lte.cell.ueScheduling.sps.
↪sassignment.downlink.rb.get(cell_name = 'abc')
```

Specifies the scheduled RB allocation for SPS DL scheduling.

param cell_name
No help available

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

set(*cell_name: str*, *number_rb: int*, *start_rb: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:SPS:SASSignment:DL:RB
driver.configure.signaling.lte.cell.ueScheduling.sps.sassignment.downlink.rb.
↪set(cell_name = 'abc', number_rb = 1, start_rb = 1)
```

Specifies the scheduled RB allocation for SPS DL scheduling.

param cell_name
No help available

param number_rb
No help available

param start_rb
No help available

6.3.4.7.2.223 SfInterval

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:SPS:SASSignment:DL:SFINterval
```

class SfIntervalCls

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

get(*cell_name: str*) → int

```
# SCPI:␣
↪[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:SPS:SASSignment:DL:SFINterval
value: int = driver.configure.signaling.lte.cell.ueScheduling.sps.sassignment.
↪downlink.sfInterval.get(cell_name = 'abc')
```

Selects the subframe periodicity for SPS DL scheduling.

param cell_name
No help available

return
sf_interval: No help available

set(*cell_name: str*, *sf_interval: int*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:SPS:SASSignment:DL:SFINterval
driver.configure.signaling.lte.cell.ueScheduling.sps.sassignment.downlink.
↪sfInterval.set(cell_name = 'abc', sf_interval = 1)
```

Selects the subframe periodicity for SPS DL scheduling.

param cell_name
No help available

param sf_interval
No help available

6.3.4.7.2.224 TbsBits

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:SPS:SASSignment:DL:TBSBits
```

class TbsBitsCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:SPS:SASSignment:DL:TBSBits
value: int = driver.configure.signaling.lte.cell.ueScheduling.sps.sassignment.
↪downlink.tbsBits.get(cell_name = 'abc')
```

Queries the transport block size in bits for SPS DL scheduling.

param cell_name
No help available

return
tbs_bits: No help available

6.3.4.7.2.225 Uplink

class UplinkCls

Uplink commands group definition. 8 total commands, 8 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.lte.cell.ueScheduling.sps.sassignment.uplink.clone()
```

Subgroups

6.3.4.7.2.226 All

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:SPS:SASSignment:UL:ALL
```

class AllCls

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

class GetStruct

Response structure. Fields:

- Sf_Interval: int: Subframe periodicity
- Number_Rb: int: No parameter help available
- Start_Rb: int: No parameter help available
- Mcs: int: No parameter help available
- Tbs_Bits: int: No parameter help available
- Ira: enums.Ira: Empty transmissions before implicit release of the UL grant
- Tic_Enable: bool: 'twoIntervalsConfig'
- Rv_Enable: bool: 'fixedRV-NonAdaptive'

class SetStruct

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

- Cell_Name: str: No parameter help available
- Sf_Interval: int: Optional setting parameter. Subframe periodicity
- Number_Rb: int: No parameter help available
- Start_Rb: int: No parameter help available

- Mcs: int: No parameter help available
- Ira: enums.Ira: Optional setting parameter. Empty transmissions before implicit release of the UL grant
- Tic_Enable: bool: Optional setting parameter. 'twoIntervalsConfig'
- Rv_Enable: bool: Optional setting parameter. 'fixedRV-NonAdaptive'

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:SPS:SASSignment:UL:ALL
value: GetStruct = driver.configure.signaling.lte.cell.ueScheduling.sps.
↪sassignment.uplink.all.get(cell_name = 'abc')
```

Configures all settings for SPS UL scheduling. A query returns the sequence <SFInterval>, <NumberRB>, <StartRB>, <MCS>, <TBSBits>, <IRA>, <TicEnable>, <RvEnable>.

param cell_name
No help available

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

set(*structure: SetStruct*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:SPS:SASSignment:UL:ALL
structure = driver.configure.signaling.lte.cell.ueScheduling.sps.sassignment.
↪uplink.all.SetStruct()
structure.Cell_Name: str = 'abc'
structure.Sf_Interval: int = 1
structure.Number_Rb: int = 1
structure.Start_Rb: int = 1
structure.Mcs: int = 1
structure.Ira: enums.Ira = enums.Ira.E2
structure.Tic_Enable: bool = False
structure.Rv_Enable: bool = False
driver.configure.signaling.lte.cell.ueScheduling.sps.sassignment.uplink.all.
↪set(structure)
```

Configures all settings for SPS UL scheduling. A query returns the sequence <SFInterval>, <NumberRB>, <StartRB>, <MCS>, <TBSBits>, <IRA>, <TicEnable>, <RvEnable>.

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

6.3.4.7.2.227 Ira

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:SPS:SASSignment:UL:IRA
```

class IraCls

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

get(*cell_name: str*) → Ira

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:SPS:SASSignment:UL:IRA
value: enums.Ira = driver.configure.signaling.lte.cell.ueScheduling.sps.
↪sassignment.uplink.ira.get(cell_name = 'abc')
```

Configures the number of empty transmissions before implicit release of the UL grant, for SPS scheduling.

param cell_name
No help available

return
ira: No help available

set(*cell_name: str*, *ira: Ira*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:SPS:SASSignment:UL:IRA
driver.configure.signaling.lte.cell.ueScheduling.sps.sassignment.uplink.ira.
↪set(cell_name = 'abc', ira = enums.Ira.E2)
```

Configures the number of empty transmissions before implicit release of the UL grant, for SPS scheduling.

param cell_name
No help available

param ira
No help available

6.3.4.7.2.228 Mcs

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:SPS:SASSignment:UL:MCS
```

class McsCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:SPS:SASSignment:UL:MCS
value: int = driver.configure.signaling.lte.cell.ueScheduling.sps.sassignment.
↪uplink.mcs.get(cell_name = 'abc')
```

Specifies the MCS index for SPS UL scheduling.

param cell_name
No help available

return
mcs: No help available

set(*cell_name: str*, *mcs: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:SPS:SASSignment:UL:MCS
driver.configure.signaling.lte.cell.ueScheduling.sps.sassignment.uplink.mcs.
↪set(cell_name = 'abc', mcs = 1)
```

Specifies the MCS index for SPS UL scheduling.

param cell_name
No help available

param mcs
No help available

6.3.4.7.2.229 Rb

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:SPS:SASSignment:UL:RB
```

class RbCls

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

class GetStruct

Response structure. Fields:

- Number_Rb: int: No parameter help available
- Start_Rb: int: No parameter help available

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:SPS:SASSignment:UL:RB
value: GetStruct = driver.configure.signaling.lte.cell.ueScheduling.sps.
↪sassignment.uplink.rb.get(cell_name = 'abc')
```

Specifies the scheduled RB allocation for SPS UL scheduling.

param cell_name
No help available

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

set(*cell_name: str*, *number_rb: int*, *start_rb: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:SPS:SASSignment:UL:RB
driver.configure.signaling.lte.cell.ueScheduling.sps.sassignment.uplink.rb.
↪set(cell_name = 'abc', number_rb = 1, start_rb = 1)
```

Specifies the scheduled RB allocation for SPS UL scheduling.

param cell_name
No help available

param number_rb
No help available

param start_rb
No help available

6.3.4.7.2.230 Rv

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:SPS:SASSignment:UL:RV
```

class RvCls

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

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:SPS:SASSignment:UL:RV
value: bool = driver.configure.signaling.lte.cell.ueScheduling.sps.sassignment.
↪uplink.rv.get(cell_name = 'abc')
```

Selects whether non-adaptive retransmissions use a fixed redundancy version (RV 0) , for SPS scheduling.

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:SPS:SASSignment:UL:RV
driver.configure.signaling.lte.cell.ueScheduling.sps.sassignment.uplink.rv.
↪set(cell_name = 'abc', enable = False)
```

Selects whether non-adaptive retransmissions use a fixed redundancy version (RV 0) , for SPS scheduling.

param cell_name
No help available

param enable
No help available

6.3.4.7.2.231 SfInterval

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:SPS:SASSignment:UL:SFINterval
```

class SfIntervalCls

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

get(*cell_name: str*) → int

```
# SCPI:␣
↪[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:SPS:SASSignment:UL:SFINterval
value: int = driver.configure.signaling.lte.cell.ueScheduling.sps.sassignment.
↪uplink.sfInterval.get(cell_name = 'abc')
```

Selects the subframe periodicity for SPS UL scheduling.

param cell_name
No help available

return
sf_interval: No help available

set(*cell_name: str*, *sf_interval: int*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:SPS:SASSignment:UL:SFINterval
driver.configure.signaling.lte.cell.ueScheduling.sps.sassignment.uplink.
↪sfInterval.set(cell_name = 'abc', sf_interval = 1)
```

Selects the subframe periodicity for SPS UL scheduling.

param cell_name
No help available

param sf_interval
No help available

6.3.4.7.2.232 TbsBits

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:SPS:SASSignment:UL:TBSBits
```

class TbsBitsCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:SPS:SASSignment:UL:TBSBits
value: int = driver.configure.signaling.lte.cell.ueScheduling.sps.sassignment.
↪uplink.tbsBits.get(cell_name = 'abc')
```

Queries the transport block size in bits for SPS UL scheduling.

param cell_name
No help available

return
tbs_bits: No help available

6.3.4.7.2.233 TiConfig

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:SPS:SASSignment:UL:TIConfig
```

class TiConfigCls

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

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:SPS:SASSignment:UL:TIConfig
value: bool = driver.configure.signaling.lte.cell.ueScheduling.sps.sassignment.
↪uplink.tiConfig.get(cell_name = 'abc')
```

Configures the parameter 'twoIntervalsConfig', signaled to the UE for scheduling type SPS in TDD mode.

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:SPS:SASSignment:UL:TIConfig
driver.configure.signaling.lte.cell.ueScheduling.sps.sassignment.uplink.
↪tiConfig.set(cell_name = 'abc', enable = False)
```

Configures the parameter 'twoIntervalsConfig', signaled to the UE for scheduling type SPS in TDD mode.

param cell_name
No help available

param enable
No help available

6.3.4.7.2.234 Uplink

class UplinkCls

Uplink commands group definition. 11 total commands, 5 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.4.7.2.235 BsrConfig

class BsrConfigCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.lte.cell.ueScheduling.uplink.bsrConfig.clone()
```

Subgroups

6.3.4.7.2.236 Mcs

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:UL:BSRConfig:MCS
```

class McsCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:UL:BSRConfig:MCS
value: int = driver.configure.signaling.lte.cell.ueScheduling.uplink.bsrConfig.
↪mcs.get(cell_name = 'abc')
```

Selects a fixed or maximum MCS value for follow BSR, MCS configuration mode FIXed or MAX.

param cell_name
No help available

return
mcs: No help available

set(*cell_name: str*, *mcs: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:UL:BSRConfig:MCS
driver.configure.signaling.lte.cell.ueScheduling.uplink.bsrConfig.mcs.set(cell_
↪name = 'abc', mcs = 1)
```

Selects a fixed or maximum MCS value for follow BSR, MCS configuration mode FIXed or MAX.

param cell_name
No help available

param mcs
No help available

6.3.4.7.2.237 McsModes

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:UL:BSRConfig:MCSModes
```

class McsModesCls

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

get(*cell_name: str*) → McsMode

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:UL:BSRConfig:MCSModes
value: enums.McsMode = driver.configure.signaling.lte.cell.ueScheduling.uplink.
↪bsrConfig.mcsModes.get(cell_name = 'abc')
```

Selects a mode for MCS configuration for follow BSR.

param cell_name
No help available

return
mode: - MAX: The maximum MCS index is configured via [CONFigure:]SIGNaling:LTE:CELL:UEScheduling:UL:BSRConfig:MCS.
- FIXed: A fixed MCS index is configured via [CONFigure:]SIGNaling:LTE:CELL:UEScheduling:UL:BSRConfig:MCS. - MMO: The maximum modulation scheme is configured via [CONFigure:]SIGNaling:LTE:CELL:UEScheduling:UL:BSRConfig:MORDer.

set(*cell_name: str*, *mode: McsMode*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:UL:BSRConfig:MCSModes
driver.configure.signaling.lte.cell.ueScheduling.uplink.bsrConfig.mcsModes.
↪set(cell_name = 'abc', mode = enums.McsMode.FIXed)
```

Selects a mode for MCS configuration for follow BSR.

param cell_name
No help available

param mode

- MAX: The maximum MCS index is configured via [CONFigure:]SIGNaling:LTE:CELL:UEScheduling:UL:BSRConfig:MCS.
- FIXed: A fixed MCS index is configured via [CONFigure:]SIGNaling:LTE:CELL:UEScheduling:UL:BSRConfig:MCS.
- MMO: The maximum modulation scheme is configured via [CONFigure:]SIGNaling:LTE:CELL:UEScheduling:UL:BSRConfig:MORDer.

6.3.4.7.2.238 Morder

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:UL:BSRConfig:MORDer
```

class MorderCls

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

get(*cell_name: str*) → ModulationOrder

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:UL:BSRConfig:MORDer
value: enums.ModulationOrder = driver.configure.signaling.lte.cell.ueScheduling.
↪uplink.bsrConfig.morder.get(cell_name = 'abc')
```

Selects the maximum modulation scheme for follow BSR, MCS configuration mode MMO.

param cell_name
No help available

return
order: No help available

set(*cell_name: str*, *order: ModulationOrder*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:UL:BSRConfig:MORDer
driver.configure.signaling.lte.cell.ueScheduling.uplink.bsrConfig.morder.
↪set(cell_name = 'abc', order = enums.ModulationOrder.Q16)
```

Selects the maximum modulation scheme for follow BSR, MCS configuration mode MMO.

param cell_name
No help available

param order
No help available

6.3.4.7.2.239 Rb

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:UL:BSRConfig:RB
```

class RbCls

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

class GetStruct

Response structure. Fields:

- Number_Rb: int: No parameter help available
- Start_Rb: int: No parameter help available

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:UL:BSRConfig:RB
value: GetStruct = driver.configure.signaling.lte.cell.ueScheduling.uplink.
↪bsrConfig.rb.get(cell_name = 'abc')
```

Defines the scheduled RB allocation for follow BSR.

param cell_name
No help available

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

set(*cell_name: str*, *number_rb: int*, *start_rb: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:UL:BSRConfig:RB
driver.configure.signaling.lte.cell.ueScheduling.uplink.bsrConfig.rb.set(cell_
↪name = 'abc', number_rb = 1, start_rb = 1)
```

Defines the scheduled RB allocation for follow BSR.

param cell_name
No help available

param number_rb
No help available

param start_rb
No help available

6.3.4.7.2.240 IgConfig

class IgConfigCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.lte.cell.ueScheduling.uplink.igConfig.clone()
```

Subgroups

6.3.4.7.2.241 Mcs

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:UL:IGConfig:MCS
```

class McsCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:UL:IGConfig:MCS
value: int = driver.configure.signaling.lte.cell.ueScheduling.uplink.igConfig.
↪mcs.get(cell_name = 'abc')
```

Selects the MCS index for the initial grant for follow SR.

param cell_name
No help available

return
mcs: No help available

set(*cell_name: str*, *mcs: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:UL:IGConfig:MCS
driver.configure.signaling.lte.cell.ueScheduling.uplink.igConfig.mcs.set(cell_
↪name = 'abc', mcs = 1)
```

Selects the MCS index for the initial grant for follow SR.

param cell_name
No help available

param mcs
No help available

6.3.4.7.2.242 Rb

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UL:IGConfig:RB
```

class RbCls

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

class GetStruct

Response structure. Fields:

- Number_Rb: int: No parameter help available
- Start_Rb: int: No parameter help available

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UL:IGConfig:RB
value: GetStruct = driver.configure.signaling.lte.cell.ueScheduling.uplink.
↪igConfig.rb.get(cell_name = 'abc')
```

Defines the scheduled RB allocation for the initial grant for follow SR.

param cell_name
No help available

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

set(*cell_name: str*, *number_rb: int*, *start_rb: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UL:IGConfig:RB
driver.configure.signaling.lte.cell.ueScheduling.uplink.igConfig.rb.set(cell_
↪name = 'abc', number_rb = 1, start_rb = 1)
```

Defines the scheduled RB allocation for the initial grant for follow SR.

param cell_name
No help available

param number_rb
No help available

param start_rb
No help available

6.3.4.7.2.243 SrcIndex

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UL:IGConfig:SRCindex
```

class SrcIndexCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UL:IGConfig:SRCindex
value: int = driver.configure.signaling.lte.cell.ueScheduling.uplink.igConfig.
↪srcIndex.get(cell_name = 'abc')
```

Configures the parameter 'sr-ConfigIndex', signaled to the UE for follow SR.

param cell_name
No help available

return
index: No help available

set(*cell_name: str*, *index: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UL:IGConfig:SRCindex
driver.configure.signaling.lte.cell.ueScheduling.uplink.igConfig.srcIndex.
↪set(cell_name = 'abc', index = 1)
```

Configures the parameter 'sr-ConfigIndex', signaled to the UE for follow SR.

param cell_name
No help available

param index
No help available

6.3.4.7.2.244 SrprIndex

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UL:IGConfig:SRPRindex
```

class SrprIndexCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UL:IGConfig:SRPRindex
value: int = driver.configure.signaling.lte.cell.ueScheduling.uplink.igConfig.
↪srprIndex.get(cell_name = 'abc')
```

Configures the parameter 'sr-PUCCH-ResourceIndex', signaled to the UE for follow SR.

param cell_name
No help available

return
index: No help available

set(*cell_name: str*, *index: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:UL:IGConfig:SRPRindex
driver.configure.signaling.lte.cell.ueScheduling.uplink.igConfig.srprIndex.
↪set(cell_name = 'abc', index = 1)
```

Configures the parameter 'sr-PUCCH-ResourceIndex', signaled to the UE for follow SR.

param cell_name
No help available

param index
No help available

6.3.4.7.2.245 McsTable

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:UL:MCSTable
```

class McsTableCls

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

get(*cell_name: str*) → McsTableD

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:UL:MCSTable
value: enums.McsTableD = driver.configure.signaling.lte.cell.ueScheduling.
↪uplink.mcsTable.get(cell_name = 'abc')
```

Selects the maximum allowed UL modulation scheme. This selection indirectly selects an MCS table for mapping of the configured MCS values to modulation schemes and TBS indices.

param cell_name
No help available

return
mcs_table: Max 16QAM, max 64QAM, max 256QAM

set(*cell_name: str*, *mcs_table: McsTableD*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:UL:MCSTable
driver.configure.signaling.lte.cell.ueScheduling.uplink.mcsTable.set(cell_name␣
↪= 'abc', mcs_table = enums.McsTableD.Q16)
```

Selects the maximum allowed UL modulation scheme. This selection indirectly selects an MCS table for mapping of the configured MCS values to modulation schemes and TBS indices.

param cell_name
No help available

param mcs_table
Max 16QAM, max 64QAM, max 256QAM

6.3.4.7.2.246 Smode

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UL:SMODe
```

class SmodeCls

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

get(*cell_name: str*) → ModeS

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UL:SMODe
value: enums.ModeS = driver.configure.signaling.lte.cell.ueScheduling.uplink.
↪smode.get(cell_name = 'abc')
```

Selects a scheduling mode for the UL.

param cell_name
No help available

return
mode: FIXed: Fixed scheduling SPS: Semi-persistent scheduling SRBSr: Follow SR/BSR UDEFined: Other scheduling mode (query only) .

set(*cell_name: str*, *mode: ModeS*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UL:SMODe
driver.configure.signaling.lte.cell.ueScheduling.uplink.smode.set(cell_name =
↪'abc', mode = enums.ModeS.FIXed)
```

Selects a scheduling mode for the UL.

param cell_name
No help available

param mode
FIXed: Fixed scheduling SPS: Semi-persistent scheduling SRBSr: Follow SR/BSR UDEFined: Other scheduling mode (query only) .

6.3.4.7.2.247 TtiBundling

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UL:TTIBundling
```

class TtiBundlingCls

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

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UL:TTIBundling
value: bool = driver.configure.signaling.lte.cell.ueScheduling.uplink.
↪ttiBundling.get(cell_name = 'abc')
```

Enables or disables TTI bundling for the uplink.

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UL:TTIBundling
driver.configure.signaling.lte.cell.ueScheduling.uplink.ttiBundling.set(cell_
↪name = 'abc', enable = False)
```

Enables or disables TTI bundling for the uplink.

param cell_name
No help available

param enable
No help available

6.3.4.7.2.248 UserDefined

class UserDefinedCls

UserDefined commands group definition. 24 total commands, 3 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.4.7.2.249 Downlink

class DownlinkCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.lte.cell.ueScheduling.userDefined.downlink.clone()
```

Subgroups

6.3.4.7.2.250 Padding

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:DL:PADDing
```

class PaddingCls

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

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:DL:PADDing
value: bool = driver.configure.signaling.lte.cell.ueScheduling.userDefined.
↪downlink.padding.get(cell_name = 'abc')
```

Activates or deactivates downlink padding at the MAC layer.

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:DL:PADDing
driver.configure.signaling.lte.cell.ueScheduling.userDefined.downlink.padding.
↪set(cell_name = 'abc', enable = False)
```

Activates or deactivates downlink padding at the MAC layer.

param cell_name
No help available

param enable
No help available

6.3.4.7.2.251 Pdcch

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:PDCCh
```

class PdcchCls

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

get(*cell_name: str*) → NoSymbols

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:PDCCh
value: enums.NoSymbols = driver.configure.signaling.lte.cell.ueScheduling.
↪userDefined.pdcch.get(cell_name = 'abc')
```

Configures the number of OFDM symbols used for the PDCCH.

param cell_name
No help available

return
no_symbols: No help available

set(*cell_name: str*, *no_symbols: NoSymbols*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:PDCCh
driver.configure.signaling.lte.cell.ueScheduling.userDefined.pdcch.set(cell_
↪name = 'abc', no_symbols = enums.NoSymbols.S1)
```

Configures the number of OFDM symbols used for the PDCCH.

param cell_name
No help available

param no_symbols
No help available

6.3.4.7.2.252 Sassignment

class SassignmentCls

Sassignment commands group definition. 22 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.lte.cell.ueScheduling.userDefined.sassignment.clone()
```

Subgroups

6.3.4.7.2.253 Downlink

class DownlinkCls

Downlink commands group definition. 12 total commands, 9 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.lte.cell.ueScheduling.userDefined.sassignment.
↪downlink.clone()
```

Subgroups

6.3.4.7.2.254 All

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:DL:ALL
```

class AllCls

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

class GetStruct

Response structure. Fields:

- Enable: List[bool]: No parameter help available
- Number_Rb: List[int]: No parameter help available
- Start_Rb: List[int]: No parameter help available
- Mcs: List[int]: No parameter help available

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:DL:ALL
value: GetStruct = driver.configure.signaling.lte.cell.ueScheduling.userDefined.
↪sassignment.downlink.all.get(cell_name = 'abc')
```

Defines the scheduled RB allocation and the MCS index for all DL subframes. The parameters are entered 10 times, so that all subframes are configured by a single command (index = subframe number 0 to 9) : <CellName>, <Enable>0, ..., <Enable>9, <NumberRB>0, ..., <NumberRB>9, <StartRB>0, ..., <StartRB>9, <MCS>0, ..., <MCS>9

param cell_name
No help available

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

set(*cell_name: str*, *enable: List[bool]*, *number_rb: List[int]*, *start_rb: List[int]*, *mcs: List[int]*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:DL:ALL
driver.configure.signaling.lte.cell.ueScheduling.userDefined.sassignment.
↪downlink.all.set(cell_name = 'abc', enable = [True, False, True], number_rb =␣
↪[1, 2, 3], start_rb = [1, 2, 3], mcs = [1, 2, 3])
```

Defines the scheduled RB allocation and the MCS index for all DL subframes. The parameters are entered 10 times, so that all subframes are configured by a single command (index = subframe number 0 to 9) : <CellName>, <Enable>0, ..., <Enable>9, <NumberRB>0, ..., <NumberRB>9, <StartRB>0, ..., <StartRB>9, <MCS>0, ..., <MCS>9

param cell_name
No help available

param enable
No help available

param number_rb
No help available

param start_rb
No help available

param mcs
No help available

6.3.4.7.2.255 Cword<Cword>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.configure.signaling.lte.cell.ueScheduling.userDefined.sassignment.downlink.
↪cword.repcap_cword_get()
driver.configure.signaling.lte.cell.ueScheduling.userDefined.sassignment.downlink.cword.
↪repcap_cword_set(repcap.Cword.Nr1)
```

class CwordCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.lte.cell.ueScheduling.userDefined.sassignment.
↪downlink.cword.clone()
```

Subgroups

6.3.4.7.2.256 Crtype

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:DL:CWORd<no>:CRTYpe
```

class CrtypeCls

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

get(*cell_name: str*, *subframe: int*, *cword=Cword.Default*) → float

```
# SCPI:␣
↪[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:DL:CWORd<no>
↪:CRTYpe
value: float = driver.configure.signaling.lte.cell.ueScheduling.userDefined.
↪sassignment.downlink.cword.crtype.get(cell_name = 'abc', subframe = 1, cword␣
↪= repcap.Cword.Default)
```

Queries the code rate for the DL subframe with the index <Subframe>, code word <no>.

param cell_name
No help available

param subframe
No help available

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

return
coderate_type: No help available

6.3.4.7.2.257 Mcs

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:DL:CWORd<no>:MCS
```

class McsCls

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

get(*cell_name: str*, *subframe: int*, *cword=Cword.Default*) → int

```
# SCPI:␣
↪[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:DL:CWORd<no>
↪:MCS
value: int = driver.configure.signaling.lte.cell.ueScheduling.userDefined.
↪sassignment.downlink.cword.mcs.get(cell_name = 'abc', subframe = 1, cword =␣
↪repcap.Cword.Default)
```

Specifies the MCS index for the DL subframe with the index <Subframe>, code word <no>.

param cell_name
No help available

param subframe
No help available

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

return
mcs: No help available

set(*cell_name: str*, *subframe: int*, *mcs: int*, *cword=Cword.Default*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:DL:CWORd<no>
↪:MCS
driver.configure.signaling.lte.cell.ueScheduling.userDefined.sassignment.
↪downlink.cword.mcs.set(cell_name = 'abc', subframe = 1, mcs = 1, cword =␣
↪repcap.Cword.Default)
```

Specifies the MCS index for the DL subframe with the index <Subframe>, code word <no>.

param cell_name
No help available

param subframe
No help available

param mcs
No help available

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

6.3.4.7.2.258 TbsBits

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:DL:CWORd<no>:TBSBits
```

class TbsBitsCls

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

get(*cell_name: str*, *subframe: int*, *cword=Cword.Default*) → int

```
# SCPI:␣
↪[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:DL:CWORd<no>
↪:TBSBits
value: int = driver.configure.signaling.lte.cell.ueScheduling.userDefined.
↪sassignment.downlink.cword.tbsBits.get(cell_name = 'abc', subframe = 1, cword␣
↪= repcap.Cword.Default)
```

Queries the transport block size bits for the DL subframe with the index <Subframe>, code word <no>.

param cell_name
No help available

param subframe
No help available

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

return
tbs_bits: No help available

6.3.4.7.2.259 TbsIndex

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:DL:CWORd<no>:TBSindex
```

class TbsIndexCls

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

get(*cell_name: str*, *subframe: int*, *cword=Cword.Default*) → int

```
# SCPI:␣
↪[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:UDEFined:SASSignment:DL:CWORd<no>
↪:TBSindex
value: int = driver.configure.signaling.lte.cell.ueScheduling.userDefined.
↪sassignment.downlink.cword.tbsIndex.get(cell_name = 'abc', subframe = 1,␣
↪cword = repcap.Cword.Default)
```

Queries the transport block size index for the DL subframe with the index <Subframe>, code word <no>.

param cell_name
No help available

param subframe
No help available

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

return
tbs_index: No help available

6.3.4.7.2.260 DciFormat

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:UDEFined:SASSignment:DL:DCIFormat
```

class DciFormatCls

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

get(*cell_name: str*, *subframe: int*) → DciFormat

```
# SCPI:␣
↪[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:UDEFined:SASSignment:DL:DCIFormat
value: enums.DciFormat = driver.configure.signaling.lte.cell.ueScheduling.
↪userDefined.sassignment.downlink.dciFormat.get(cell_name = 'abc', subframe =␣
↪1)
```

Defines the DCI format for the DL subframe with the index <Subframe>.

param cell_name
No help available

param subframe
No help available

return
dci_format: No help available

set(*cell_name: str*, *subframe: int*, *dci_format: DciFormat*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:DL:DCIFormat
driver.configure.signaling.lte.cell.ueScheduling.userDefined.sassignment.
↪downlink.dciFormat.set(cell_name = 'abc', subframe = 1, dci_format = enums.
↪DciFormat.D0)
```

Defines the DCI format for the DL subframe with the index <Subframe>.

param cell_name
No help available

param subframe
No help available

param dci_format
No help available

6.3.4.7.2.261 Enable

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:DL:ENABle
```

class EnableCls

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

get(*cell_name: str*, *subframe: int*) → bool

```
# SCPI:␣
↪[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:DL:ENABle
value: bool = driver.configure.signaling.lte.cell.ueScheduling.userDefined.
↪sassignment.downlink.enable.get(cell_name = 'abc', subframe = 1)
```

Enables or disables scheduling of the DL subframe with the index <Subframe>.

param cell_name
No help available

param subframe
No help available

return
enable: No help available

set(*cell_name: str*, *subframe: int*, *enable: bool*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:DL:ENABle
driver.configure.signaling.lte.cell.ueScheduling.userDefined.sassignment.
↪downlink.enable.set(cell_name = 'abc', subframe = 1, enable = False)
```

Enables or disables scheduling of the DL subframe with the index <Subframe>.

param cell_name
No help available

param subframe
No help available

param enable
No help available

6.3.4.7.2.262 McsTable

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:DL:MCSTable
```

class McsTableCls

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

get(*cell_name: str*) → McsTable

```
# SCPI:␣
↪[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:DL:MCSTable
value: enums.McsTable = driver.configure.signaling.lte.cell.ueScheduling.
↪userDefined.sassignment.downlink.mcsTable.get(cell_name = 'abc')
```

Selects the maximum allowed DL modulation scheme. This selection indirectly selects an MCS table for mapping of the configured MCS values to modulation schemes and TBS indices.

param cell_name
No help available

return
mcs_table: Max 64QAM, max 256QAM, max 1024QAM

set(*cell_name: str*, *mcs_table: McsTable*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:DL:MCSTable
driver.configure.signaling.lte.cell.ueScheduling.userDefined.sassignment.
↪downlink.mcsTable.set(cell_name = 'abc', mcs_table = enums.McsTable.Q1K)
```

Selects the maximum allowed DL modulation scheme. This selection indirectly selects an MCS table for mapping of the configured MCS values to modulation schemes and TBS indices.

param cell_name
No help available

param mcs_table
Max 64QAM, max 256QAM, max 1024QAM

6.3.4.7.2.263 PdcchFormat

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:DL:PDCChformat
```

class PdcchFormatCls

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

get(*cell_name: str*, *subframe: int*) → PdcchFormat

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:DL:PDCChformat
value: enums.PdcchFormat = driver.configure.signaling.lte.cell.ueScheduling.userDefined.sassignment.downlink.pdcchFormat.get(cell_name = 'abc', subframe = 1)
```

Queries the number of CCEs used for transmission of the PDCCH, for the DL subframe with the index <Subframe>.

param cell_name
No help available

param subframe
No help available

return
pdcch_format: No help available

6.3.4.7.2.264 Rb

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:DL:RB
```

class RbCls

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

class GetStruct

Response structure. Fields:

- Number_Rb: int: No parameter help available
- Start_Rb: int: No parameter help available

get(*cell_name: str*, *subframe: int*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:DL:RB
value: GetStruct = driver.configure.signaling.lte.cell.ueScheduling.userDefined.sassignment.downlink.rb.get(cell_name = 'abc', subframe = 1)
```

Specifies the scheduled RB allocation for the DL subframe with the index <Subframe>.

param cell_name
No help available

param subframe
No help available

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

set(*cell_name: str*, *subframe: int*, *number_rb: int*, *start_rb: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:DL:RB
driver.configure.signaling.lte.cell.ueScheduling.userDefined.sassignment.
↪downlink.rb.set(cell_name = 'abc', subframe = 1, number_rb = 1, start_rb = 1)
```

Specifies the scheduled RB allocation for the DL subframe with the index <Subframe>.

param cell_name
No help available

param subframe
No help available

param number_rb
No help available

param start_rb
No help available

6.3.4.7.2.265 Riv

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:DL:RIV
```

class RivCls

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

get(*cell_name: str*, *subframe: int*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:DL:RIV
value: int = driver.configure.signaling.lte.cell.ueScheduling.userDefined.
↪sassignment.downlink.riv.get(cell_name = 'abc', subframe = 1)
```

Defines the resource indication value (RIV) for the DL subframe with the index <Subframe>.

param cell_name
No help available

param subframe
No help available

return
riv: No help available

set(*cell_name: str*, *subframe: int*, *riv: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:DL:RIV
driver.configure.signaling.lte.cell.ueScheduling.userDefined.sassignment.
↪downlink.riv.set(cell_name = 'abc', subframe = 1, riv = 1)
```

Defines the resource indication value (RIV) for the DL subframe with the index <Subframe>.

param cell_name
No help available

param subframe
No help available

param riv
No help available

6.3.4.7.2.266 Tmode

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:DL:TMODe
```

class TmodeCls

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

get(*cell_name: str*) → Tmode

```
# SCPI:␣
↪[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:DL:TMODe
value: enums.Tmode = driver.configure.signaling.lte.cell.ueScheduling.
↪userDefined.sassignment.downlink.tmode.get(cell_name = 'abc')
```

Selects the DL transmission mode.

param cell_name
No help available

return
tmode: No help available

set(*cell_name: str*, *tmode: Tmode*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:DL:TMODe
driver.configure.signaling.lte.cell.ueScheduling.userDefined.sassignment.
↪downlink.tmode.set(cell_name = 'abc', tmode = enums.Tmode.TM1)
```

Selects the DL transmission mode.

param cell_name
No help available

param tmode
No help available

6.3.4.7.2.267 Uplink

class UplinkCls

Uplink commands group definition. 10 total commands, 7 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.lte.cell.ueScheduling.userDefined.sassignment.uplink.
↪clone()
```

Subgroups

6.3.4.7.2.268 All

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:UDEFined:SASSignment:UL:ALL
```

class AllCls

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

class GetStruct

Response structure. Fields:

- Enable: List[bool]: No parameter help available
- Number_Rb: List[int]: No parameter help available
- Start_Rb: List[int]: No parameter help available
- Mcs: List[int]: No parameter help available

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:UDEFined:SASSignment:UL:ALL
value: GetStruct = driver.configure.signaling.lte.cell.ueScheduling.userDefined.
↪sassignment.uplink.all.get(cell_name = 'abc')
```

Defines the scheduled RB allocation and the MCS index for all UL subframes. The parameters are entered 10 times, so that all subframes are configured by a single command (index = subframe number 0 to 9) : <CellName>, <Enable>0, …, <Enable>9, <NumberRB>0, …, <NumberRB>9, <StartRB>0, …, <StartRB>9, <MCS>0, …, <MCS>9

param cell_name
No help available

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

set(*cell_name: str*, *enable: List[bool]*, *number_rb: List[int]*, *start_rb: List[int]*, *mcs: List[int]*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UEScheduling:UDEFined:SASSignment:UL:ALL
driver.configure.signaling.lte.cell.ueScheduling.userDefined.sassignment.uplink.
↪all.set(cell_name = 'abc', enable = [True, False, True], number_rb = [1, 2,␣
↪3], start_rb = [1, 2, 3], mcs = [1, 2, 3])
```

Defines the scheduled RB allocation and the MCS index for all UL subframes. The parameters are entered 10 times, so that all subframes are configured by a single command (index = subframe number 0 to 9) : <CellName>, <Enable>0, ..., <Enable>9, <NumberRB>0, ..., <NumberRB>9, <StartRB>0, ..., <StartRB>9, <MCS>0, ..., <MCS>9

param cell_name
No help available

param enable
No help available

param number_rb
No help available

param start_rb
No help available

param mcs
No help available

6.3.4.7.2.269 Cword<Cword>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.configure.signaling.lte.cell.ueScheduling.userDefined.sassignment.uplink.
↪cword.repcap_cword_get()
driver.configure.signaling.lte.cell.ueScheduling.userDefined.sassignment.uplink.cword.
↪repcap_cword_set(repcap.Cword.Nr1)
```

class CwordCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.lte.cell.ueScheduling.userDefined.sassignment.uplink.
↪cword.clone()
```

Subgroups

6.3.4.7.2.270 Crtype

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:UL:CWORd<no>:CRTYpe
```

class CrtypeCls

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

get(*cell_name: str*, *subframe: int*, *cword=Cword.Default*) → float

```
# SCPI:␣
↪[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:UL:CWORd<no>
↪:CRTYpe
value: float = driver.configure.signaling.lte.cell.ueScheduling.userDefined.
↪sassignment.uplink.cword.crtype.get(cell_name = 'abc', subframe = 1, cword =␣
↪repcap.Cword.Default)
```

Queries the code rate for the UL subframe with the index <Subframe>, code word <no>.

param cell_name
No help available

param subframe
No help available

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

return
coderate_type: No help available

6.3.4.7.2.271 Mcs

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:UL:CWORd<no>:MCS
```

class McsCls

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

get(*cell_name: str*, *subframe: int*, *cword=Cword.Default*) → int

```
# SCPI:␣
↪[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:UL:CWORd<no>
↪:MCS
value: int = driver.configure.signaling.lte.cell.ueScheduling.userDefined.
↪sassignment.uplink.cword.mcs.get(cell_name = 'abc', subframe = 1, cword =␣
↪repcap.Cword.Default)
```

Specifies the MCS index for the UL subframe with the index <Subframe>, code word <no>.

param cell_name
No help available

param subframe
No help available

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

return
mcs: No help available

set(*cell_name: str*, *subframe: int*, *mcs: int*, *cword=Cword.Default*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:UDEFined:SASSignment:UL:CWORd<no>
↪:MCS
driver.configure.signaling.lte.cell.ueScheduling.userDefined.sassignment.uplink.
↪cword.mcs.set(cell_name = 'abc', subframe = 1, mcs = 1, cword = repcap.Cword.
↪Default)
```

Specifies the MCS index for the UL subframe with the index <Subframe>, code word <no>.

param cell_name
No help available

param subframe
No help available

param mcs
No help available

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

6.3.4.7.2.272 TbsBits

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:UDEFined:SASSignment:UL:CWORd<no>:TBSBits
```

class TbsBitsCls

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

get(*cell_name: str*, *subframe: int*, *cword=Cword.Default*) → int

```
# SCPI:␣
↪[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:UDEFined:SASSignment:UL:CWORd<no>
↪:TBSBits
value: int = driver.configure.signaling.lte.cell.ueScheduling.userDefined.
↪sassignment.uplink.cword.tbsBits.get(cell_name = 'abc', subframe = 1, cword =␣
↪repcap.Cword.Default)
```

Queries the transport block size bits for the UL subframe with the index <Subframe>, code word <no>.

param cell_name
No help available

param subframe
No help available

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

return
tbs_bits: No help available

6.3.4.7.2.273 TbsIndex

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:UL:CWORd<no>:TBSindex
```

class TbsIndexCls

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

get(*cell_name: str*, *subframe: int*, *cword=Cword.Default*) → int

```
# SCPI:␣
↪[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:UL:CWORd<no>
↪:TBSindex
value: int = driver.configure.signaling.lte.cell.ueScheduling.userDefined.
↪sassignment.uplink.cword.tbsIndex.get(cell_name = 'abc', subframe = 1, cword␣
↪= repcap.Cword.Default)
```

Queries the transport block size index for the UL subframe with the index <Subframe>, code word <no>.

param cell_name
No help available

param subframe
No help available

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

return
tbs_index: No help available

6.3.4.7.2.274 DciFormat

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:UL:DCIFormat
```

class DciFormatCls

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

get(*cell_name: str*, *subframe: int*) → DciFormat

```
# SCPI:␣
↪[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:UL:DCIFormat
value: enums.DciFormat = driver.configure.signaling.lte.cell.ueScheduling.
↪userDefined.sassignment.uplink.dciFormat.get(cell_name = 'abc', subframe = 1)
```

Defines the DCI format for the UL subframe with the index <Subframe>.

param cell_name
No help available

param subframe
No help available

return
dci_format: No help available

set(*cell_name: str*, *subframe: int*, *dci_format: DciFormat*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:UL:DCIFormat
driver.configure.signaling.lte.cell.ueScheduling.userDefined.sassignment.uplink.
↪dciFormat.set(cell_name = 'abc', subframe = 1, dci_format = enums.DciFormat.
↪D0)
```

Defines the DCI format for the UL subframe with the index <Subframe>.

param cell_name
No help available

param subframe
No help available

param dci_format
No help available

6.3.4.7.2.275 Enable

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:UL:ENABle
```

class EnableCls

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

get(*cell_name: str*, *subframe: int*) → bool

```
# SCPI:␣
↪[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:UDEFined:SASSignment:UL:ENABle
value: bool = driver.configure.signaling.lte.cell.ueScheduling.userDefined.
↪sassignment.uplink.enable.get(cell_name = 'abc', subframe = 1)
```

Enables or disables scheduling of the UL subframe with the index <Subframe>.

param cell_name
No help available

param subframe
No help available

return
enable: No help available

set(*cell_name: str*, *subframe: int*, *enable: bool*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:UDEFined:SASSignment:UL:ENABle
driver.configure.signaling.lte.cell.ueScheduling.userDefined.sassignment.uplink.
↪enable.set(cell_name = 'abc', subframe = 1, enable = False)
```

Enables or disables scheduling of the UL subframe with the index <Subframe>.

param cell_name
No help available

param subframe
No help available

param enable
No help available

6.3.4.7.2.276 PdcchFormat

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:UDEFined:SASSignment:UL:PDCChformat
```

class PdcchFormatCls

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

get(*cell_name: str*, *subframe: int*) → PdcchFormat

```
# SCPI:␣
↪[CONFigure]:SIGNaling:LTE:CELL:UEScheduling:UDEFined:SASSignment:UL:PDCChformat
value: enums.PdcchFormat = driver.configure.signaling.lte.cell.ueScheduling.
↪userDefined.sassignment.uplink.pdcchFormat.get(cell_name = 'abc', subframe =␣
↪1)
```

Queries the number of CCEs used for transmission of the PDCCH, for the UL subframe with the index <Subframe>.

param cell_name
No help available

param subframe
No help available

return
pdcch_format: No help available

6.3.4.7.2.277 Rb

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:UL:RB
```

class RbCls

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

class GetStruct

Response structure. Fields:

- Number_Rb: int: No parameter help available
- Start_Rb: int: No parameter help available

get(*cell_name: str*, *subframe: int*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:UL:RB
value: GetStruct = driver.configure.signaling.lte.cell.ueScheduling.userDefined.
↪sassignment.uplink.rb.get(cell_name = 'abc', subframe = 1)
```

Specifies the scheduled RB allocation for the UL subframe with the index <Subframe>.

param cell_name
No help available

param subframe
No help available

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

set(*cell_name: str*, *subframe: int*, *number_rb: int*, *start_rb: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:UL:RB
driver.configure.signaling.lte.cell.ueScheduling.userDefined.sassignment.uplink.
↪rb.set(cell_name = 'abc', subframe = 1, number_rb = 1, start_rb = 1)
```

Specifies the scheduled RB allocation for the UL subframe with the index <Subframe>.

param cell_name
No help available

param subframe
No help available

param number_rb
No help available

param start_rb
No help available

6.3.4.7.2.278 Riv

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:UL:RIV
```

class RivCls

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

get(*cell_name: str*, *subframe: int*) → int

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:UL:RIV
value: int = driver.configure.signaling.lte.cell.ueScheduling.userDefined.
↪sassignment.uplink.riv.get(cell_name = 'abc', subframe = 1)
```

Defines the resource indication value (RIV) for the UL subframe with the index <Subframe>.

param cell_name
No help available

param subframe
No help available

return
riv: No help available

set(*cell_name: str*, *subframe: int*, *riv: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:UESCheduling:UDEFined:SASSignment:UL:RIV
driver.configure.signaling.lte.cell.ueScheduling.userDefined.sassignment.uplink.
↪riv.set(cell_name = 'abc', subframe = 1, riv = 1)
```

Defines the resource indication value (RIV) for the UL subframe with the index <Subframe>.

param cell_name
No help available

param subframe
No help available

param riv
No help available

6.3.4.7.2.279 UlIndication

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:CELL:ULINdication
```

class UlIndicationCls

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

get(*cell_name: str*) → UlIndication

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:ULINdication
value: enums.UlIndication = driver.configure.signaling.lte.cell.ulIndication.
↪get(cell_name = 'abc')
```

Configures whether the optional parameter 'upperLayerIndication' is signaled to the UE in SIB 2 or not.

param cell_name
No help available

return
indication: AUTO: Signaled if EN-DC is active for the LTE cell. AON: Always signaled. AOFF: Never signaled.

set(*cell_name: str*, *indication: UlIndication*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:CELL:ULINdication
driver.configure.signaling.lte.cell.ulIndication.set(cell_name = 'abc',␣
↪indication = enums.UlIndication.AOFF)
```

Configures whether the optional parameter 'upperLayerIndication' is signaled to the UE in SIB 2 or not.

param cell_name
No help available

param indication
AUTO: Signaled if EN-DC is active for the LTE cell. AON: Always signaled. AOFF: Never signaled.

6.3.4.7.3 Ncell

class NcellCls

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

Cloning the Group

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

Subgroups

6.3.4.7.3.1 Thresholds

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:NCELl:THResholds
```

class ThresholdsCls

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

class GetStruct

Response structure. Fields:

- Threshold_Low: float: Threshold ThreshX Low ('ThreshX, LowP') .
- Threshold_High: float: Threshold ThreshX High ('ThreshX, HighP') .

get(*cell_name: str*, *ncell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:LTE:NCELl:THResholds
value: GetStruct = driver.configure.signaling.lte.ncell.thresholds.get(cell_
↪name = 'abc', ncell_name = 'abc')
```

Configures reselection thresholds for an entry in the neighbor cell list of an LTE or NR cell.

param cell_name
Serving LTE or NR cell via which the neighbor cell list is broadcasted.

param ncell_name
Neighbor cell

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

set(*cell_name: str*, *ncell_name: str*, *threshold_low: float*, *threshold_high: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:NCELl:THResholds
driver.configure.signaling.lte.ncell.thresholds.set(cell_name = 'abc', ncell_
↪name = 'abc', threshold_low = 1.0, threshold_high = 1.0)
```

Configures reselection thresholds for an entry in the neighbor cell list of an LTE or NR cell.

param cell_name
Serving LTE or NR cell via which the neighbor cell list is broadcasted.

param ncell_name
Neighbor cell

param threshold_low
Threshold ThreshX Low ('ThreshX, LowP') .

param threshold_high
Threshold ThreshX High ('ThreshX, HighP') .

6.3.4.7.4 Ue

class UeCls

Ue commands group definition. 4 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.4.7.4.1 Bearer

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:UE:BEARer
```

class BearerCls

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

class GetStruct

Response structure. Fields:

- Qci: enums.Qi: Value of the quality of service class identifier. Values defined in 3GPP TS 23.203, table 6.1.7. The GUI shows the designation of each value.
- Max_Dl_Bitrate: int: Maximum DL bit rate allowed in the network.
- Max_Ul_Bitrate: int: Maximum UL bit rate allowed in the network.
- Gtd_Dl_Bitrate: int: DL bit rate guaranteed by the network for the bearer.
- Gtd_Ul_Bitrate: int: UL bit rate guaranteed by the network for the bearer.

class SetStruct

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

- Ue_Id: str: No parameter help available
- Bearer_Id: int: No parameter help available
- Qci: enums.Qi: Optional setting parameter. Value of the quality of service class identifier. Values defined in 3GPP TS 23.203, table 6.1.7. The GUI shows the designation of each value.
- Max_Dl_Bitrate: int: Optional setting parameter. Maximum DL bit rate allowed in the network.
- Max_Ul_Bitrate: int: Optional setting parameter. Maximum UL bit rate allowed in the network.
- Gtd_Dl_Bitrate: int: Optional setting parameter. DL bit rate guaranteed by the network for the bearer.
- Gtd_Ul_Bitrate: int: Optional setting parameter. UL bit rate guaranteed by the network for the bearer.

get(*ue_id: str*, *bearer_id: int*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:LTE:UE:BEARer
value: GetStruct = driver.configure.signaling.lte.ue.bearer.get(ue_id = 'abc',␣
↪bearer_id = 1)
```

Configures the existing bearer with the <BearerId>.

param ue_id
No help available

param bearer_id
No help available

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

set(*structure: SetStruct*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:UE:BEARer
structure = driver.configure.signaling.lte.ue.bearer.SetStruct()
structure.Ue_Id: str = 'abc'
structure.Bearer_Id: int = 1
structure.Qci: enums.Qi = enums.Qi.Q1
structure.Max_Dl_Bitrate: int = 1
structure.Max_Ul_Bitrate: int = 1
structure.Gtd_Dl_Bitrate: int = 1
structure.Gtd_Ul_Bitrate: int = 1
driver.configure.signaling.lte.ue.bearer.set(structure)
```

Configures the existing bearer with the <BearerId>.

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

6.3.4.7.4.2 Nsa

SCPI Commands :

```
[CONFigure]:SIGNaling:LTE:UE:NSA:ACTivate
[CONFigure]:SIGNaling:LTE:UE:NSA:DEACtivate
```

class NsaCls

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

class ActivateStruct

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

- Ue_Id: str: Optional setting parameter. For future use. Enter any value if you want to use optional parameters.
- Linked_Bearer_Id: int: Optional setting parameter. ID of the default bearer to which the dedicated bearer is linked. To get a list of all default bearer IDs, see [CMDLINKRESOLVED Catalog.Signaling.Lte.Ue.Dbearer#get_ CMDLINKRESOLVED].
- Data_Flow: enums.DataFlow: Optional setting parameter. Configures the user data flow for the dedicated bearer. MCGSplit: MCG split bearer, with traffic split in the eNB SCG: SCG via gNB, no traffic split SCGSplit: SCG split bearer, with traffic split in the gNB
- Traffic_Dist: float: Optional setting parameter. For MCGSplit and SCGSplit. A numeric value defines the percentage of the data to be transferred via the interface eNB - UE. The remainder is transferred via the interface gNB - UE. AUTO configures the traffic distribution automatically and dynamically, depending on the load in the eNB path.
- Qci: enums.Qi: Optional setting parameter. Value of the quality of service class identifier. Values defined in 3GPP TS 23.203, table 6.1.7. The GUI shows the designation of each value.
- Max_Dl_Bitrate: int: Optional setting parameter. Maximum DL bit rate allowed in the network.

- Max_Ul_Bitrate: int: Optional setting parameter. Maximum UL bit rate allowed in the network.
- Gtd_Dl_Bitrate: int: Optional setting parameter. DL bit rate guaranteed by the network for the bearer.
- Gtd_Ul_Bitrate: int: Optional setting parameter. UL bit rate guaranteed by the network for the bearer.
- Pscell: str: Optional setting parameter. Name of the NR cell for which you want to activate EN-DC.
- Rlc_Mode: enums.RlcMode: Optional setting parameter. RLC mode ACK: acknowledged UACK: unacknowledged

activate(*structure: ActivateStruct*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:UE:NSA:ACTivate
structure = driver.configure.signaling.lte.ue.nsa.ActivateStruct()
structure.Ue_Id: str = 'abc'
structure.Linked_Bearer_Id: int = 1
structure.Data_Flow: enums.DataFlow = enums.DataFlow.MCG
structure.Traffic_Dist: float = 1.0
structure.Qci: enums.Qi = enums.Qi.Q1
structure.Max_Dl_Bitrate: int = 1
structure.Max_Ul_Bitrate: int = 1
structure.Gtd_Dl_Bitrate: int = 1
structure.Gtd_Ul_Bitrate: int = 1
structure.Pscell: str = 'abc'
structure.Rlc_Mode: enums.RlcMode = enums.RlcMode.ACK
driver.configure.signaling.lte.ue.nsa.activate(structure)
```

Activates the EN-DC mode and establishes a dedicated bearer.

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

deactivate(*ue_id: str = None*, *bearer_id: int = None*, *esm_cause: EsmCause = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:UE:NSA:DEACtivate
driver.configure.signaling.lte.ue.nsa.deactivate(ue_id = 'abc', bearer_id = 1,␣
↪esm_cause = enums.EsmCause.C100)
```

Deactivates the EN-DC mode and releases a dedicated bearer.

param ue_id
For future use. Enter any value if you want to use optional parameters.

param bearer_id
ID of the dedicated bearer to be released. To get a list of all dedicated bearer IDs, see method **RsCMX_Signaling.Catalog.Signaling.Lte.Ue.Bearer.get_**.

param esm_cause
Release cause to be sent. Values defined in 3GPP TS 24.301, chapter 9.9.4.4.

Cloning the Group

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

Subgroups

6.3.4.7.4.3 Resume

SCPI Command :

```
[CONFigure]:SIGNaling:LTE:UE:NSA:RESume
```

class ResumeCls

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

set(*ue_id: str = None*, *bearer_id: List[int] = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:LTE:UE:NSA:RESume
driver.configure.signaling.lte.ue.nsa.resume.set(ue_id = 'abc', bearer_id = [1,␣
↪2, 3])
```

Resumes the EN-DC mode after an error resulting in a suspended EN-DC mode.

param ue_id
For future use. Enter any value if you want to use optional parameters.

param bearer_id
ID of the dedicated bearer. To get a list of all dedicated bearer IDs, see method **RsCMX_Signaling.Catalog.Signaling.Lte.Ue.Dbearer.get_**.

6.3.4.8 Measurement

class MeasurementCls

Measurement commands group definition. 17 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.4.8.1 Bler

SCPI Commands :

```
[CONFigure]:SIGNaling:MEASurement:BLER:REPetition
[CONFigure]:SIGNaling:MEASurement:BLER:SMAVerage
```

class BlerCls

Bler commands group definition. 7 total commands, 3 Subgroups, 2 group commands

get_repetition() → Repeat

```
# SCPI: [CONFigure]:SIGNaling:MEASurement:BLER:REPetition
value: enums.Repeat = driver.configure.signaling.measurement.bler.get_
↪repetition()
```

Specifies whether the measurement is stopped after a single shot or repeated continuously. After setting SINGleshot, also configure a stop condition. After setting CONTinuous, do not configure a stop condition. See [CONFigure:]SIGNaling:MEASurement:BLER:SCONdition.

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

get_sm_average() → float

```
# SCPI: [CONFigure]:SIGNaling:MEASurement:BLER:SMAVerage
value: float = driver.configure.signaling.measurement.bler.get_sm_average()
```

Configures the number of samples to be considered for CONTinuous measurements (last n samples) .

return
samples: No help available

set_repetition(*repetition: Repeat*) → None

```
# SCPI: [CONFigure]:SIGNaling:MEASurement:BLER:REPetition
driver.configure.signaling.measurement.bler.set_repetition(repetition = enums.
↪Repeat.CONTinuous)
```

Specifies whether the measurement is stopped after a single shot or repeated continuously. After setting SINGleshot, also configure a stop condition. After setting CONTinuous, do not configure a stop condition. See [CONFigure:]SIGNaling:MEASurement:BLER:SCONdition.

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

set_sm_average(*samples: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:MEASurement:BLER:SMAVerage
driver.configure.signaling.measurement.bler.set_sm_average(samples = 1.0)
```

Configures the number of samples to be considered for CONTinuous measurements (last n samples) .

param samples
No help available

Cloning the Group

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

Subgroups

6.3.4.8.1.1 Cmeasure

SCPI Commands :

```
[CONFigure]:SIGNaling:MEASurement:BLER:CMEasure:CELLs
[CONFigure]:SIGNaling:MEASurement:BLER:CMEasure
```

class CmeasureCls

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

get_cells() → List[str]

```
# SCPI: [CONFigure]:SIGNaling:MEASurement:BLER:CMEasure:CELLs
value: List[str] = driver.configure.signaling.measurement.bler.cmeasure.get_
↪cells()
```

Selects the cells to be evaluated by the BLER measurement.

return
cell_name: Comma-separated list of cells

get_value() → CellsTypeToMeasure

```
# SCPI: [CONFigure]:SIGNaling:MEASurement:BLER:CMEasure
value: enums.CellsTypeToMeasure = driver.configure.signaling.measurement.bler.
↪cmeasure.get_value()
```

Selects the scope of the BLER measurement.

return
cells_to_measure: CGRoup: Measure certain cell groups, selected via [CONFigure:]SIGNaling:MEASurement:BLER:CMEasure:CGRoup. CELLs: Measure certain cells, selected via [CONFigure:]SIGNaling:MEASurement:BLER:CMEasure:CELLs.

set_cells(*cell_name: List[str]*) → None

```
# SCPI: [CONFigure]:SIGNaling:MEASurement:BLER:CMEasure:CELLs
driver.configure.signaling.measurement.bler.cmeasure.set_cells(cell_name = [
↪'abc1', 'abc2', 'abc3'])
```

Selects the cells to be evaluated by the BLER measurement.

param cell_name
Comma-separated list of cells

set_value(*cells_to_measure: CellsTypeToMeasure*) → None

```
# SCPI: [CONFigure]:SIGNaling:MEASurement:BLER:CMEasure
driver.configure.signaling.measurement.bler.cmeasure.set_value(cells_to_measure␣
↪= enums.CellsTypeToMeasure.CELLs)
```

Selects the scope of the BLER measurement.

param cells_to_measure
CGRoup: Measure certain cell groups, selected via [CONFigure:]SIGNaling:MEASurement:BLER:CMEasure:CGRoup. CELLs: Measure certain cells, selected via [CONFigure:]SIGNaling:MEASurement:BLER:CMEasure:CELLs.

Cloning the Group

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

Subgroups

6.3.4.8.1.2 Cgroup

SCPI Command :

```
[CONFigure]:SIGNaling:MEASurement:BLER:CMEasure:CGRoup
```

class CgroupCls

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

class CgroupStruct

Response structure. Fields:

- Mcg: bool: No parameter help available
- Scg: bool: No parameter help available

get() → CgroupStruct

```
# SCPI: [CONFigure]:SIGNaling:MEASurement:BLER:CMEasure:CGRoup
value: CgroupStruct = driver.configure.signaling.measurement.bler.cmeasure.
↪cgroup.get()
```

Selects the cell groups to be evaluated by the BLER measurement.

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

set(*mcg: bool*, *scg: bool = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:MEASurement:BLER:CMEasure:CGRoup
driver.configure.signaling.measurement.bler.cmeasure.cgroup.set(mcg = False,␣
↪scg = False)
```

Selects the cell groups to be evaluated by the BLER measurement.

param mcg
No help available

param scg
No help available

6.3.4.8.1.3 Enable

SCPI Command :

```
[CONFigure]:SIGNaling:MEASurement:BLER:ENABle
```

class EnableCls

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

class EnableStruct

Response structure. Fields:

- Dl: bool: Measure DL BLER results (ON) or not (OFF) .
- Ul: bool: Measure UL BLER results (ON) or not (OFF) .

get() → EnableStruct

```
# SCPI: [CONFigure]:SIGNaling:MEASurement:BLER:ENABle
value: EnableStruct = driver.configure.signaling.measurement.bler.enable.get()
```

Enables or disables the measurement of DL BLER and UL BLER results. This command affects BLER measurements started via a remote command. BLER measurements started via the GUI always deliver DL BLER and UL BLER results.

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

set(*dl: bool*, *ul: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:MEASurement:BLER:ENABle
driver.configure.signaling.measurement.bler.enable.set(dl = False, ul = False)
```

Enables or disables the measurement of DL BLER and UL BLER results. This command affects BLER measurements started via a remote command. BLER measurements started via the GUI always deliver DL BLER and UL BLER results.

param dl
Measure DL BLER results (ON) or not (OFF) .

param ul
Measure UL BLER results (ON) or not (OFF) .

6.3.4.8.1.4 Scondition

SCPI Command :

```
[CONFigure]:SIGNaling:MEASurement:BLER:SCONdition
```

class SconditionCls

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

class SconditionStruct

Response structure. Fields:

- Stop_Condition: enums.StopCondition: SAMPles: number of samples reached for at least one cell TIME: measurement duration reached CONFidence: confidence BLER verdict derived
- Value: float: For SAMPles, number of samples. For TIME, measurement duration. For CONFidence, number of samples applied beyond the maximum value defined by 3GPP (2466 without CA, 1003 per CC with CA) .

get() → SconditionStruct

```
# SCPI: [CONFigure]:SIGNaling:MEASurement:BLER:SCONdition
value: SconditionStruct = driver.configure.signaling.measurement.bler.
↪scondition.get()
```

Defines a stop condition for single-shot BLER measurements.

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

set(*stop_condition: StopCondition*, *value: float = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:MEASurement:BLER:SCONdition
driver.configure.signaling.measurement.bler.scondition.set(stop_condition =␣
↪enums.StopCondition.CONFidence, value = 1.0)
```

Defines a stop condition for single-shot BLER measurements.

param stop_condition
SAMPles: number of samples reached for at least one cell TIME: measurement duration reached CONFidence: confidence BLER verdict derived

param value
For SAMPles, number of samples. For TIME, measurement duration. For CONFidence, number of samples applied beyond the maximum value defined by 3GPP (2466 without CA, 1003 per CC with CA) .

6.3.4.8.2 UeReport

SCPI Commands :

```
[CONFigure]:SIGNaling:MEASurement:UEReport:ENABle
[CONFigure]:SIGNaling:MEASurement:UEReport:RINTerval
```

class UeReportCls

UeReport commands group definition. 10 total commands, 2 Subgroups, 2 group commands

get_enable() → CellsToMeasure

```
# SCPI: [CONFigure]:SIGNaling:MEASurement:UEReport:ENABle
value: enums.CellsToMeasure = driver.configure.signaling.measurement.ueReport.
↪get_enable()
```

Selects whether the UE must send measurement reports and for which type of serving cell.

return
cells_to_measure: OFF: no serving cell measurement reports ALL: reporting for any serving cell LTE: reporting for LTE serving cell NRADio: reporting for NR serving cell

get_rinterval() → ReportInterval

```
# SCPI: [CONFigure]:SIGNaling:MEASurement:UEReport:RINTerval
value: enums.ReportInterval = driver.configure.signaling.measurement.ueReport.
↪get_rinterval()
```

Configures the interval between two consecutive serving cell measurement reports.

return
report_interval: I1 to I5: 120 ms, 240 ms, 480 ms, 640 ms, 1024 ms I6 to I10: 2048 ms, 5120 ms, 10240 ms, 20480 ms, 40960 ms I11 to I14: 1 min, 6 min, 12 min, 30 min

set_enable(*cells_to_measure: CellsToMeasure*) → None

```
# SCPI: [CONFigure]:SIGNaling:MEASurement:UEReport:ENABle
driver.configure.signaling.measurement.ueReport.set_enable(cells_to_measure =␣
↪enums.CellsToMeasure.ALL)
```

Selects whether the UE must send measurement reports and for which type of serving cell.

param cells_to_measure
OFF: no serving cell measurement reports ALL: reporting for any serving cell LTE: reporting for LTE serving cell NRADio: reporting for NR serving cell

set_rinterval(*report_interval: ReportInterval*) → None

```
# SCPI: [CONFigure]:SIGNaling:MEASurement:UEReport:RINTerval
driver.configure.signaling.measurement.ueReport.set_rinterval(report_interval =␣
↪enums.ReportInterval.I1)
```

Configures the interval between two consecutive serving cell measurement reports.

param report_interval
I1 to I5: 120 ms, 240 ms, 480 ms, 640 ms, 1024 ms I6 to I10: 2048 ms, 5120 ms, 10240 ms, 20480 ms, 40960 ms I11 to I14: 1 min, 6 min, 12 min, 30 min

Cloning the Group

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

Subgroups

6.3.4.8.2.1 Ncell

SCPI Commands :

```
[CONFigure]:SIGNaling:MEASurement:UEReport:NCELl:ENABle
[CONFigure]:SIGNaling:MEASurement:UEReport:NCELl:RINTerval
```

class NcellCls

Ncell commands group definition. 7 total commands, 1 Subgroups, 2 group commands

get_enable() → bool

```
# SCPI: [CONFigure]:SIGNaling:MEASurement:UEReport:NCELl:ENABle
value: bool = driver.configure.signaling.measurement.ueReport.ncell.get_enable()
```

No command help available

return
enable: No help available

get_rinterval() → ReportInterval

```
# SCPI: [CONFigure]:SIGNaling:MEASurement:UEReport:NCELl:RINTerval
value: enums.ReportInterval = driver.configure.signaling.measurement.ueReport.
↪ncell.get_rinterval()
```

Configures the interval between two consecutive neighbor cell measurement reports. Applies only to <Type> = CNETwork, see [CONFigure:]SIGNaling:MEASurement:UEReport:NCELl:RESult:TYPE

return
report_interval: I1 to I5: 120 ms, 240 ms, 480 ms, 640 ms, 1024 ms I6 to I10: 2048 ms, 5120 ms, 10240 ms, 20480 ms, 40960 ms I11 to I14: 1 min, 6 min, 12 min, 30 min

set_enable(*enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:MEASurement:UEReport:NCELl:ENABle
driver.configure.signaling.measurement.ueReport.ncell.set_enable(enable = False)
```

No command help available

param enable
No help available

set_rinterval(*report_interval: ReportInterval*) → None

```
# SCPI: [CONFigure]:SIGNaling:MEASurement:UEReport:NCELl:RINTerval
driver.configure.signaling.measurement.ueReport.ncell.set_rinterval(report_
↪interval = enums.ReportInterval.I1)
```

Configures the interval between two consecutive neighbor cell measurement reports. Applies only to <Type> = CNETwork, see [CONFigure:]SIGNaling:MEASurement:UEReport:NCELl:RESult:TYPE

param report_interval
I1 to I5: 120 ms, 240 ms, 480 ms, 640 ms, 1024 ms I6 to I10: 2048 ms, 5120 ms, 10240 ms, 20480 ms, 40960 ms I11 to I14: 1 min, 6 min, 12 min, 30 min

Cloning the Group

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

Subgroups

6.3.4.8.2.2 Result

SCPI Commands :

```
[CONFigure]:SIGNaling:MEASurement:UEReport:NCELl:RESult:TYPE
[CONFigure]:SIGNaling:MEASurement:UEReport:NCELl:RESult:CNETwork
[CONFigure]:SIGNaling:MEASurement:UEReport:NCELl:RESult:SIB
[CONFigure]:SIGNaling:MEASurement:UEReport:NCELl:RESult:NCList
```

class ResultCls

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

get_cnetwork() → CellsToMeasure

```
# SCPI: [CONFigure]:SIGNaling:MEASurement:UEReport:NCELl:RESult:CNETwork
value: enums.CellsToMeasure = driver.configure.signaling.measurement.ueReport.
↪ncell.result.get_cnetwork()
```

Selects whether the UE must send measurement reports and for which neighbor cells. Applies only to <Type> = CNETwork, see [CONFigure:]SIGNaling:MEASurement:UEReport:NCELl:RESult:TYPE

return
cells_to_measure: OFF: no neighbor cell measurement reports ALL: reporting for any neighbor cell LTE: LTE neighbor cells with licensed bands NRADio: NR neighbor cells LLAA: LTE neighbor cells with licensed or LAA bands LAA: LTE neighbor cells with LAA bands

get_nc_list() → List[str]

```
# SCPI: [CONFigure]:SIGNaling:MEASurement:UEReport:NCELl:RESult:NCList
value: List[str] = driver.configure.signaling.measurement.ueReport.ncell.result.
↪get_nc_list()
```

No command help available

return
cell_name: No help available

get_sib() → NcellsToMeasure

```
# SCPI: [CONFigure]:SIGNaling:MEASurement:UEReport:NCELl:RESult:SIB
value: enums.NcellsToMeasure = driver.configure.signaling.measurement.ueReport.
↪ncell.result.get_sib()
```

Selects whether the UE must send measurement reports and for which neighbor cells. Applies only to <Type> = SIB, see [CONFigure:]SIGNaling:MEASurement:UEReport:NCELl:RESult:TYPE.

return
ncells_to_measure: OFF: no neighbor cell measurement reports ALL: reporting for all SIB neighbor cells IAFRequency: reporting intra-frequency SIB neighbor cells IFRequency: reporting inter-frequency SIB neighbor cells IRAT: reporting for inter-RAT SIB neighbor cells

get_type_py() → NeighborCellType

```
# SCPI: [CONFigure]:SIGNaling:MEASurement:UEReport:NCELl:RESult:TYPE
value: enums.NeighborCellType = driver.configure.signaling.measurement.ueReport.
↪ncell.result.get_type_py()
```

Selects the type of neighbor cell list to be used for neighbor cell measurements.

return
type_py: CNETwork: all created cells except the serving cell SIB: SIB neighbor cell list (configured neighbors of serving cell) NCList: for future use

set_cnetwork(*cells_to_measure: CellsToMeasure*) → None

```
# SCPI: [CONFigure]:SIGNaling:MEASurement:UEReport:NCELl:RESult:CNETwork
driver.configure.signaling.measurement.ueReport.ncell.result.set_cnetwork(cells_
↪to_measure = enums.CellsToMeasure.ALL)
```

Selects whether the UE must send measurement reports and for which neighbor cells. Applies only to <Type> = CNETwork, see [CONFigure:]SIGNaling:MEASurement:UEReport:NCELl:RESult:TYPE

param cells_to_measure
OFF: no neighbor cell measurement reports ALL: reporting for any neighbor cell LTE: LTE neighbor cells with licensed bands NRADio: NR neighbor cells LLAA: LTE neighbor cells with licensed or LAA bands LAA: LTE neighbor cells with LAA bands

set_nc_list(*cell_name: List[str]*) → None

```
# SCPI: [CONFigure]:SIGNaling:MEASurement:UEReport:NCELl:RESult:NCList
driver.configure.signaling.measurement.ueReport.ncell.result.set_nc_list(cell_
↪name = ['abc1', 'abc2', 'abc3'])
```

No command help available

param cell_name
No help available

set_sib(*ncells_to_measure: NcellsToMeasure*) → None

```
# SCPI: [CONFigure]:SIGNaling:MEASurement:UEReport:NCELl:RESult:SIB
driver.configure.signaling.measurement.ueReport.ncell.result.set_sib(ncells_to_
↪measure = enums.NcellsToMeasure.ALL)
```

Selects whether the UE must send measurement reports and for which neighbor cells. Applies only to <Type> = SIB, see [CONFigure:]SIGNaling:MEASurement:UEReport:NCELl:RESult:TYPE.

param ncells_to_measure
OFF: no neighbor cell measurement reports ALL: reporting for all SIB neighbor cells IAFRequency: reporting intra-frequency SIB neighbor cells IFRequency: reporting inter-frequency SIB neighbor cells IRAT: reporting for inter-RAT SIB neighbor cells

set_type_py(*type_py: NeighborCellType*) → None

```
# SCPI: [CONFigure]:SIGNaling:MEASurement:UEReport:NCELl:RESult:TYPE
driver.configure.signaling.measurement.ueReport.ncell.result.set_type_py(type_
↪py = enums.NeighborCellType.CNETwork)
```

Selects the type of neighbor cell list to be used for neighbor cell measurements.

param type_py
CNETwork: all created cells except the serving cell SIB: SIB neighbor cell list (configured neighbors of serving cell) NCList: for future use

Cloning the Group

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

Subgroups

6.3.4.8.2.3 Enable

SCPI Command :

```
[CONFigure]:SIGNaling:MEASurement:UEReport:NCELl:RESult:ENABle
```

class EnableCls

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

class EnableStruct

Response structure. Fields:

- Rsrp: bool: No parameter help available
- Rsrq: bool: No parameter help available
- Rssi_Nr: bool: No parameter help available

get() → EnableStruct

```
# SCPI: [CONFigure]:SIGNaling:MEASurement:UEReport:NCELl:RESult:ENABle
value: EnableStruct = driver.configure.signaling.measurement.ueReport.ncell.
↪result.enable.get()
```

Selects the quantities to be reported by the UE for neighbor cell measurements. Applies only to <Type> = CNETwork, see [CONFigure:]SIGNaling:MEASurement:UEReport:NCELl:RESult:TYPE

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

set(*rsrp: bool*, *rsrq: bool*, *rssi_nr: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:MEASurement:UEReport:NCELl:RESult:ENABle
driver.configure.signaling.measurement.ueReport.ncell.result.enable.set(rsrp =␣
↪False, rsrq = False, rssi_nr = False)
```

Selects the quantities to be reported by the UE for neighbor cell measurements. Applies only to <Type> = CNETwork, see [CONFigure:]SIGNaling:MEASurement:UEReport:NCELl:RESult:TYPE

param rsrp
No help available

param rsrq
No help available

param rssi_nr
No help available

6.3.4.8.2.4 Result

class ResultCls

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

Cloning the Group

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

Subgroups

6.3.4.8.2.5 Enable

SCPI Command :

```
[CONFigure]:SIGNaling:MEASurement:UEReport:RESult:ENABle
```

class EnableCls

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

class EnableStruct

Response structure. Fields:

- Rsrp: bool: No parameter help available
- Rsrq: bool: No parameter help available
- Rssi_Nr: bool: No parameter help available

get() → EnableStruct

```
# SCPI: [CONFigure]:SIGNaling:MEASurement:UEReport:RESult:ENABle
value: EnableStruct = driver.configure.signaling.measurement.ueReport.result.
↪enable.get()
```

Selects the quantities to be reported by the UE for serving cell measurements.

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

set(*rsrp: bool*, *rsrq: bool*, *rssi_nr: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:MEASurement:UEReport:RESult:ENABle
driver.configure.signaling.measurement.ueReport.result.enable.set(rsrp = False,␣
↪rsrq = False, rssi_nr = False)
```

Selects the quantities to be reported by the UE for serving cell measurements.

param rsrp
No help available

param rsrq
No help available

param rssi_nr
No help available

6.3.4.9 Nbehavior

SCPI Command :

```
[CONFigure]:SIGNaling:NBEHavior:DITimer
```

class NbehaviorCls

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

get_di_timer() → float

```
# SCPI: [CONFigure]:SIGNaling:NBEHavior:DITimer
value: float or bool = driver.configure.signaling.nbehavior.get_di_timer()
```

No command help available

return
timer: (float or boolean) No help available

set_di_timer(*timer: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:NBEHavior:DITimer
driver.configure.signaling.nbehavior.set_di_timer(timer = 1.0)
```

No command help available

param timer
(float or boolean) No help available

6.3.4.10 Nradio

class NradioCls

Nradio commands group definition. 542 total commands, 3 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.4.10.1 Ca

class CaCls

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

Cloning the Group

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

Subgroups

6.3.4.10.1.1 Dormancy

class DormancyCls

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

Cloning the Group

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

Subgroups

6.3.4.10.1.2 Switch

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CA:DORMancy:SWITch
```

class SwitchCls

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

class GetStruct

Response structure. Fields:

- Dormant: bool: Target dormancy state dormant (ON) or non-dormant (OFF) .
- **Indication_Mode: enums.IndicationMode:**
  - AUTO: Send the indication with the next scheduled DCI with format 0_1 or 1_1 or 2_6.
  - WAT1: Send the indication with the next scheduled DCI with format 0_1 or 1_1, within the active time of the DRX cycle.
  - WAT2: Send the indication with the next scheduled DCI with format 1_1, within the active time of the DRX cycle.
  - OATime: Send the indication with the next scheduled DCI with format 2_6, outside the active time of the DRX cycle.

get(*cell_group_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CA:DORMancy:SWITch
value: GetStruct = driver.configure.signaling.nradio.ca.dormancy.switch.
↪get(cell_group_name = 'abc')
```

Switches the dormancy state and selects the DCI format used to transport the dormancy indication to the UE.

param cell_group_name
No help available

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

set(*cell_group_name: str*, *dormant: bool*, *indication_mode: IndicationMode*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CA:DORMancy:SWITch
driver.configure.signaling.nradio.ca.dormancy.switch.set(cell_group_name = 'abc
↪', dormant = False, indication_mode = enums.IndicationMode.AUTO)
```

Switches the dormancy state and selects the DCI format used to transport the dormancy indication to the UE.

param cell_group_name
No help available

param dormant
Target dormancy state dormant (ON) or non-dormant (OFF) .

param indication_mode

- AUTO: Send the indication with the next scheduled DCI with format 0_1 or 1_1 or 2_6.
- WAT1: Send the indication with the next scheduled DCI with format 0_1 or 1_1, within the active time of the DRX cycle.
- WAT2: Send the indication with the next scheduled DCI with format 1_1, within the active time of the DRX cycle.
- OATime: Send the indication with the next scheduled DCI with format 2_6, outside the active time of the DRX cycle.

6.3.4.10.1.3 Scell

class ScellCls

Scell commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.4.10.1.4 Activation

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CA:SCELl:ACTivation
```

class ActivationCls

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

get(*cell_name: List[str]*) → List[bool]

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CA:SCELl:ACTivation
value: List[bool] = driver.configure.signaling.nradio.ca.scell.activation.
↪get(cell_name = ['abc1', 'abc2', 'abc3'])
```

Triggers the manual MAC activation or MAC deactivation for an SCell or several SCells. A query returns the current MAC activation state for an SCell.

param cell_name
No help available

return
activation: ON: Activate MAC (setting) / MAC is active (query) . OFF: Deactivate MAC (setting) / MAC is inactive (query) .

set(*cell_name: List[str]*, *activation: List[bool] = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CA:SCELl:ACTivation
driver.configure.signaling.nradio.ca.scell.activation.set(cell_name = ['abc1',
↪'abc2', 'abc3'], activation = [True, False, True])
```

Triggers the manual MAC activation or MAC deactivation for an SCell or several SCells. A query returns the current MAC activation state for an SCell.

param cell_name
No help available

param activation
ON: Activate MAC (setting) / MAC is active (query) . OFF: Deactivate MAC (setting) / MAC is inactive (query) .

6.3.4.10.1.5 Dormancy

class DormancyCls

Dormancy commands group definition. 4 total commands, 3 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.4.10.1.6 Dbwp

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CA:SCELl:DORMancy:DBWP
```

class DbwpCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CA:SCELl:DORMancy:DBWP
value: int = driver.configure.signaling.nradio.ca.scell.dormancy.dbwp.get(cell_
↪name = 'abc')
```

Selects the target DL BWP for switching the cell state to dormant (‘dormantBWP-Id’) .

param cell_name
No help available

return
bwp: IBWP: initial BWP integer: BWP ID

set(*cell_name: str*, *bwp: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CA:SCELl:DORMancy:DBWP
driver.configure.signaling.nradio.ca.scell.dormancy.dbwp.set(cell_name = 'abc',␣
↪bwp = 1)
```

Selects the target DL BWP for switching the cell state to dormant (‘dormantBWP-Id’) .

param cell_name
No help available

param bwp
IBWP: initial BWP integer: BWP ID

6.3.4.10.1.7 Enable

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CA:SCELl:DORMancy:ENABle
```

class EnableCls

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

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CA:SCELl:DORMancy:ENABle
value: bool = driver.configure.signaling.nradio.ca.scell.dormancy.enable.
↪get(cell_name = 'abc')
```

Enables signaling dormancy settings for the cell to the UE.

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CA:SCELl:DORMancy:ENABle
driver.configure.signaling.nradio.ca.scell.dormancy.enable.set(cell_name = 'abc
↪', enable = False)
```

Enables signaling dormancy settings for the cell to the UE.

param cell_name
No help available

param enable
No help available

6.3.4.10.1.8 NdBwp

class NdBwpCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.ca.scell.dormancy.ndBwp.clone()
```

Subgroups

6.3.4.10.1.9 OaTime

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CA:SCELl:DORMancy:NDBWp:OATime
```

class OaTimeCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CA:SCELl:DORMancy:NDBWp:OATime
value: int = driver.configure.signaling.nradio.ca.scell.dormancy.ndBwp.oaTime.
↪get(cell_name = 'abc')
```

Selects the target DL BWP for switching to non-dormant and sending the dormancy indication outside of the active time of the DRX cycle ('firstOutsideActiveTimeBWP-Id') .

param cell_name
No help available

return
bwp: IBWP: initial BWP integer: BWP ID

set(*cell_name: str*, *bwp: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CA:SCELl:DORMancy:NDBWp:OATime
driver.configure.signaling.nradio.ca.scell.dormancy.ndBwp.oaTime.set(cell_name␣
↪= 'abc', bwp = 1)
```

Selects the target DL BWP for switching to non-dormant and sending the dormancy indication outside of the active time of the DRX cycle ('firstOutsideActiveTimeBWP-Id') .

param cell_name
No help available

param bwp
IBWP: initial BWP integer: BWP ID

6.3.4.10.1.10 WaTime

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CA:SCELl:DORMancy:NDBWp:WATime
```

class WaTimeCls

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

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CA:SCELl:DORMancy:NDBWp:WATime
value: int = driver.configure.signaling.nradio.ca.scell.dormancy.ndBwp.waTime.
↪get(cell_name = 'abc')
```

Selects the target DL BWP for switching to non-dormant and sending the dormancy indication within the active time of the DRX cycle ('firstWithinActiveTimeBWP-Id') .

param cell_name
No help available

return
bwp: IBWP: initial BWP integer: BWP ID

set(*cell_name: str*, *bwp: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CA:SCELl:DORMancy:NDBWp:WATime
driver.configure.signaling.nradio.ca.scell.dormancy.ndBwp.waTime.set(cell_name␣
↪= 'abc', bwp = 1)
```

Selects the target DL BWP for switching to non-dormant and sending the dormancy indication within the active time of the DRX cycle ('firstWithinActiveTimeBWP-Id') .

param cell_name
No help available

param bwp
IBWP: initial BWP integer: BWP ID

6.3.4.10.1.11 Mac

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CA:SCELl:MAC
```

class MacCls

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

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CA:SCELl:MAC
value: bool = driver.configure.signaling.nradio.ca.scell.mac.get(cell_name =
↪'abc')
```

Enables or disables the retransmission of the MAC activation CE message for SCells if the activation is not acknowledged by the UE. Modifying this setting for a cell modifies it also for all other cells of the cell group.

param cell_name
No help available

return
activation: No help available

set(*cell_name: str*, *activation: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CA:SCELl:MAC
driver.configure.signaling.nradio.ca.scell.mac.set(cell_name = 'abc',␣
↪activation = False)
```

Enables or disables the retransmission of the MAC activation CE message for SCells if the activation is not acknowledged by the UE. Modifying this setting for a cell modifies it also for all other cells of the cell group.

param cell_name
No help available

param activation
No help available

6.3.4.10.1.12 Uplink

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CA:SCELl:UL
```

class UplinkCls

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

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CA:SCELl:UL
value: bool = driver.configure.signaling.nradio.ca.scell.uplink.get(cell_name =
↪'abc')
```

Enables or disables the uplink of an SCell.

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CA:SCELl:UL
driver.configure.signaling.nradio.ca.scell.uplink.set(cell_name = 'abc', enable␣
↪= False)
```

Enables or disables the uplink of an SCell.

param cell_name
No help available

param enable
No help available

6.3.4.10.2 Cell

class CellCls

Cell commands group definition. 532 total commands, 38 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.4.10.2.1 Alayout

class AlayoutCls

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

Cloning the Group

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

Subgroups

6.3.4.10.2.2 Layout

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:ALAYout:LAYout
```

class LayoutCls

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

get(*cell_name: str*) → AntennaLayout

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:ALAYout:LAYout
value: enums.AntennaLayout = driver.configure.signaling.nradio.cell.alayout.
↪layout.get(cell_name = 'abc')
```

Configures the layout of the CSI antenna port array.

param cell_name
No help available

return
antenna_layout: TX2 for 1 or 2 CSI-RS antenna ports Nab for 2*a*b antenna ports, a=N1, b=N2

set(*cell_name: str*, *antenna_layout: AntennaLayout*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:ALAYout:LAYout
driver.configure.signaling.nradio.cell.alayout.layout.set(cell_name = 'abc',␣
↪antenna_layout = enums.AntennaLayout.N121)
```

Configures the layout of the CSI antenna port array.

param cell_name
No help available

param antenna_layout
TX2 for 1 or 2 CSI-RS antenna ports Nab for 2*a*b antenna ports, a=N1, b=N2

6.3.4.10.2.3 Asn

class AsnCls

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

Cloning the Group

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

Subgroups

6.3.4.10.2.4 Mib

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:ASN:MIB
```

class MibCls

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

set(*cell_name: str*, *message: str*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:ASN:MIB
driver.configure.signaling.nradio.cell.asn.mib.set(cell_name = 'abc', message =
↪'abc')
```

No command help available

param cell_name
No help available

param message
No help available

6.3.4.10.2.5 Sib1

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:ASN:SIB1
```

class Sib1Cls

Sib1 commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*cell_name: str*, *message: str*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:ASN:SIB1
driver.configure.signaling.nradio.cell.asn.sib1.set(cell_name = 'abc', message␣
↪= 'abc')
```

No command help available

param cell_name
No help available

param message
No help available

6.3.4.10.2.6 Barred

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BARRed
```

class BarredCls

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

class GetStruct

Response structure. Fields:

- Enable: bool: ON: cell barred OFF: cell not barred
- Ifri_Present: bool: 'intraFreqReselectionRedCap-r17' = 'allowed'
- One_Rx_Barred: bool: 'cellBarredRedCap1Rx-r17' = 'barred'
- Two_Rx_Barred: bool: 'cellBarredRedCap2Rx-r17' = 'barred'
- Half_Duplex: bool: 'halfDuplexRedCapAllowed-r17' = 'false'

class SetStruct

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

- Cell_Name: str: No parameter help available
- Enable: bool: ON: cell barred OFF: cell not barred
- Ifri_Present: bool: Optional setting parameter. 'intraFreqReselectionRedCap-r17' = 'allowed'
- One_Rx_Barred: bool: Optional setting parameter. 'cellBarredRedCap1Rx-r17' = 'barred'
- Two_Rx_Barred: bool: Optional setting parameter. 'cellBarredRedCap2Rx-r17' = 'barred'
- Half_Duplex: bool: Optional setting parameter. 'halfDuplexRedCapAllowed-r17' = 'false'

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BARRed
value: GetStruct = driver.configure.signaling.nradio.cell.barred.get(cell_name
↪= 'abc')
```

Specifies whether the cell is barred.

param cell_name
No help available

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

set(*structure: SetStruct*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BARRed
structure = driver.configure.signaling.nradio.cell.barred.SetStruct()
structure.Cell_Name: str = 'abc'
structure.Enable: bool = False
structure.Ifri_Present: bool = False
structure.One_Rx_Barred: bool = False
structure.Two_Rx_Barred: bool = False
structure.Half_Duplex: bool = False
driver.configure.signaling.nradio.cell.barred.set(structure)
```

Specifies whether the cell is barred.

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

6.3.4.10.2.7 BbCombining

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BBCombining
```

class BbCombiningCls

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

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BBCombining
value: bool = driver.configure.signaling.nradio.cell.bbCombining.get(cell_name
↪= 'abc')
```

Allows baseband combining for the cell.

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BBCombining
driver.configure.signaling.nradio.cell.bbCombining.set(cell_name = 'abc',␣
↪enable = False)
```

Allows baseband combining for the cell.

param cell_name
No help available

param enable
No help available

6.3.4.10.2.8 Beam

class BeamCls

Beam commands group definition. 7 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.4.10.2.9 Following

class FollowingCls

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

Cloning the Group

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

Subgroups

6.3.4.10.2.10 All

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BEAM:FOLLowing:ALL
```

class AllCls

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

class GetStruct

Response structure. Fields:

- Enable: bool: Enables signaling of a change of the DL beam to the UE.
- Follow_Mode: enums.ModeBfollow: OFF: No beam following AUTO: Beam selection based on UE measurement reports BLOCk: Beamlock configuration via BeamLockMode and Index
- Beam_Lock_Mode: enums.Mode: Type of value to be used for target selection. SSBBeam: SSB beam index BINDex: beam index CSIRs: NZP CSI-RS resource ID
- Index: int: Value of the type BeamLockMode, e.g. an SBB beam index value.

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BEAM:FOLLowing:ALL
value: GetStruct = driver.configure.signaling.nradio.cell.beam.following.all.
↪get(cell_name = 'abc')
```

Configures all settings for DL beam following in a single command.

param cell_name
No help available

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

set(*cell_name: str*, *enable: bool*, *follow_mode: ModeBfollow = None*, *beam_lock_mode: Mode = None*, *index: int = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BEAM:FOLLowing:ALL
driver.configure.signaling.nradio.cell.beam.following.all.set(cell_name = 'abc',
↪ enable = False, follow_mode = enums.ModeBfollow.AUTO, beam_lock_mode = enums.
↪Mode.BINDex, index = 1)
```

Configures all settings for DL beam following in a single command.

param cell_name
No help available

param enable
Enables signaling of a change of the DL beam to the UE.

param follow_mode
OFF: No beam following AUTO: Beam selection based on UE measurement reports BLOCk: Beamlock configuration via BeamLockMode and Index

param beam_lock_mode
Type of value to be used for target selection. SSBBeam: SSB beam index BINDex: beam index CSIRs: NZP CSI-RS resource ID

param index
Value of the type BeamLockMode, e.g. an SBB beam index value.

6.3.4.10.2.11 Block

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BEAM:FOLLowing:BLOCk
```

class BlockCls

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

class GetStruct

Response structure. Fields:

- Mode: enums.Mode: Type of value to be used for target selection. SSBBeam: SSB beam index BINDex: beam index CSIRs: NZP CSI-RS resource ID
- Index: int: Value of the type Mode, e.g. an SBB beam index value.

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BEAM:FOLLowing:BLOCk
value: GetStruct = driver.configure.signaling.nradio.cell.beam.following.block.
↪get(cell_name = 'abc')
```

Selects a beamlock target.

param cell_name
No help available

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

set(*cell_name: str*, *mode: Mode*, *index: int = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BEAM:FOLLowing:BLOCk
driver.configure.signaling.nradio.cell.beam.following.block.set(cell_name = 'abc
↪', mode = enums.Mode.BINDex, index = 1)
```

Selects a beamlock target.

param cell_name
No help available

param mode
Type of value to be used for target selection. SSBBeam: SSB beam index BINDex: beam index CSIRs: NZP CSI-RS resource ID

param index
Value of the type Mode, e.g. an SBB beam index value.

6.3.4.10.2.12 Fmode

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BEAM:FOLLowing:FMODe
```

class FmodeCls

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

get(*cell_name: str*) → ModeBfollow

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BEAM:FOLLowing:FMODe
value: enums.ModeBfollow = driver.configure.signaling.nradio.cell.beam.
↪following.fmode.get(cell_name = 'abc')
```

Selects a mode for the DL beam following.

param cell_name
No help available

return
mode: OFF: No beam following AUTO: Beam selection based on UE measurement reports BLOCk: Beamlock configuration via [CONFigure:]SIGNaling:NRADio:CELL:BEAM:FOLLowing:BLOCk

set(*cell_name: str*, *mode: ModeBfollow*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BEAM:FOLLowing:FMODe
driver.configure.signaling.nradio.cell.beam.following.fmode.set(cell_name = 'abc
↪', mode = enums.ModeBfollow.AUTO)
```

Selects a mode for the DL beam following.

param cell_name
No help available

param mode
OFF: No beam following AUTO: Beam selection based on UE measurement reports BLOCk: Beamlock configuration via [CONFigure:]SIGNaling:NRADio:CELL:BEAM:FOLLowing:BLOCk

6.3.4.10.2.13 IbChange

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BEAM:FOLLowing:IBCHange
```

class IbChangeCls

IbChange commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BEAM:FOLLowing:IBCHange
value: bool = driver.configure.signaling.nradio.cell.beam.following.ibChange.
↪get(cell_name = 'abc')
```

Selects whether a change of the DL beam is signaled to the UE.

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BEAM:FOLLowing:IBCHange
driver.configure.signaling.nradio.cell.beam.following.ibChange.set(cell_name =
↪'abc', enable = False)
```

Selects whether a change of the DL beam is signaled to the UE.

param cell_name
No help available

param enable
No help available

6.3.4.10.2.14 Frecovery

class FrecoveryCls

Frecovery commands group definition. 3 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.beam.frecovery.clone()
```

Subgroups

6.3.4.10.2.15 Mode

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BEAM:FRECovery:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → ModeFrecovery

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BEAM:FRECovery:MODE
value: enums.ModeFrecovery = driver.configure.signaling.nradio.cell.beam.
↪frecovery.mode.get(cell_name = 'abc')
```

Selects a mode for configuration of the candidate list for beam failure recovery.

param cell_name
No help available

return
mode: - OFF: No candidate list, no recovery. - AUTO: The active beam is the only candidate. - UDEFined: Configuration via [CONFigure:]SIGNaling:NRADio:CELL:BEAM:FRECovery:UDEFined:SSB and [CONFigure:]SIGNaling:NRADio:CELL:BEAM:FRECovery:UDEFined:CSIRs.

set(*cell_name: str*, *mode: ModeFrecovery*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BEAM:FRECovery:MODE
driver.configure.signaling.nradio.cell.beam.frecovery.mode.set(cell_name = 'abc
↪', mode = enums.ModeFrecovery.AUTO)
```

Selects a mode for configuration of the candidate list for beam failure recovery.

param cell_name
No help available

param mode

- OFF: No candidate list, no recovery.
- AUTO: The active beam is the only candidate.
- UDEFined: Configuration via [CONFigure:]SIGNaling:NRADio:CELL:BEAM:FRECovery:UDEFined:SSB and [CONFigure:]SIGNaling:NRADio:CELL:BEAM:FRECovery:UDEFined:CSIRs.

6.3.4.10.2.16 UserDefined

class UserDefinedCls

UserDefined commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.beam.frecovery.userDefined.clone()
```

Subgroups

6.3.4.10.2.17 Csirs

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BEAM:FRECovery:UDEFined:CSIRs
```

class CsirsCls

Csirs commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Nzb_Id: List[int]: NZP CSI-RS resource index
- Candidate: List[bool]: The CSI-RS beam with index NZBId is a candidate (ON) or not (OFF) .

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BEAM:FRECovery:UDEFined:CSIRs
value: GetStruct = driver.configure.signaling.nradio.cell.beam.frecovery.
↪userDefined.csirs.get(cell_name = 'abc')
```

Specifies the CSI-RS part of a user-defined candidate list for beam failure recovery. See also [CONFigure:]SIGNaling:NRADio:CELL:BEAM:FRECovery:MODE.

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *nzb_id: List[int]*, *candidate: List[bool]*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BEAM:FRECovery:UDEFined:CSIRs
driver.configure.signaling.nradio.cell.beam.frecovery.userDefined.csirs.
↪set(cell_name = 'abc', nzb_id = [1, 2, 3], candidate = [True, False, True])
```

Specifies the CSI-RS part of a user-defined candidate list for beam failure recovery. See also [CONFigure:]SIGNaling:NRADio:CELL:BEAM:FRECovery:MODE.

param cell_name
No help available

param nzb_id
NZP CSI-RS resource index

param candidate
The CSI-RS beam with index NZBId is a candidate (ON) or not (OFF) .

6.3.4.10.2.18 Ssb

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BEAM:FRECovery:UDEFined:SSB
```

class SsbCls

Ssb commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Ssb_Block: List[int]: SSB index
- Candidate: List[bool]: The SSB with index SSBBlock is a candidate (ON) or not (OFF) .

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BEAM:FRECovery:UDEFined:SSB
value: GetStruct = driver.configure.signaling.nradio.cell.beam.frecovery.
↪userDefined.ssb.get(cell_name = 'abc')
```

Specifies the SSB beam portion of a user-defined candidate list for beam failure recovery. See also [CONFigure:]SIGNaling:NRADio:CELL:BEAM:FRECovery:MODE.

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *ssb_block: List[int]*, *candidate: List[bool]*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BEAM:FRECovery:UDEFined:SSB
driver.configure.signaling.nradio.cell.beam.frecovery.userDefined.ssb.set(cell_
↪name = 'abc', ssb_block = [1, 2, 3], candidate = [True, False, True])
```

Specifies the SSB beam portion of a user-defined candidate list for beam failure recovery. See also [CONFigure:]SIGNaling:NRADio:CELL:BEAM:FRECovery:MODE.

param cell_name
No help available

param ssb_block
SSB index

param candidate
The SSB with index SSBBlock is a candidate (ON) or not (OFF) .

6.3.4.10.2.19 Beams

class BeamsCls

Beams commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.beams.clone()
```

Subgroups

6.3.4.10.2.20 Ap3Trigger

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BEAMs:AP3Trigger
```

class Ap3TriggerCls

Ap3Trigger commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BEAMs:AP3Trigger
value: bool = driver.configure.signaling.nradio.cell.beams.ap3Trigger.get(cell_
↪name = 'abc')
```

Configures automatic triggering of the P-3 process upon update of the active CSI-RS beam.

param cell_name
No help available

return
auto_p_3_trigger: Automatic triggering OFF or ON.

set(*cell_name: str*, *auto_p_3_trigger: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BEAMs:AP3Trigger
driver.configure.signaling.nradio.cell.beams.ap3Trigger.set(cell_name = 'abc',␣
↪auto_p_3_trigger = False)
```

Configures automatic triggering of the P-3 process upon update of the active CSI-RS beam.

param cell_name
No help available

param auto_p_3_trigger
Automatic triggering OFF or ON.

6.3.4.10.2.21 BeamConfig

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BEAMs:BEAMconfig
```

class BeamConfigCls

BeamConfig commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Resource_Id: int: ID of the CSI-RS beam
- Aoa: enums.AoaB: Angle of arrival of the CSI-RS beam
- Phase: float: Phase of the CSI-RS beam
- Relative_Power: float: Power difference of the CSI-RS beam
- Po_Vs_Sss: int: Power offset of a CSI-RS RE to an SSS RE

class SetStruct

Structure for setting input parameters. Contains optional setting parameters. Fields:

- Cell_Name: str: No parameter help available
- Resource_Id: int: ID of the CSI-RS beam
- Aoa: enums.AoaB: Optional setting parameter. Angle of arrival of the CSI-RS beam
- Phase: float: Optional setting parameter. Phase of the CSI-RS beam
- Relative_Power: float: Optional setting parameter. Power difference of the CSI-RS beam
- Po_Vs_Sss: int: Optional setting parameter. Power offset of a CSI-RS RE to an SSS RE

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BEAMs:BEAMconfig
value: GetStruct = driver.configure.signaling.nradio.cell.beams.beamConfig.
↪get(cell_name = 'abc')
```

Configures the NZP CSI-RS beams in the active SSB beam.

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*structure: SetStruct*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BEAMs:BEAMconfig
structure = driver.configure.signaling.nradio.cell.beams.beamConfig.SetStruct()
structure.Cell_Name: str = 'abc'
structure.Resource_Id: int = 1
structure.Aoa: enums.AoaB = enums.AoaB.AOA1
structure.Phase: float = 1.0
structure.Relative_Power: float = 1.0
structure.Po_Vs_Sss: int = 1
driver.configure.signaling.nradio.cell.beams.beamConfig.set(structure)
```

Configures the NZP CSI-RS beams in the active SSB beam.

param structure
for set value, see the help for SetStruct structure arguments.

6.3.4.10.2.22 Mtrigger

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BEAMs:MTRigger
```

class MtriggerCls

Mtrigger commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*cell_name: str*, *trigger: BeamsTrigger*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BEAMs:MTRigger
driver.configure.signaling.nradio.cell.beams.mtrigger.set(cell_name = 'abc',␣
↪trigger = enums.BeamsTrigger.ACTive)
```

Triggers the P-3 process on the active CSI-RS beam or on a selected CSI-RS beam.

param cell_name
No help available

param trigger
Selects the CSI-RS beam.

6.3.4.10.2.23 NbBeams

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BEAMs:NBBeams
```

class NbBeamsCls

NbBeams commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BEAMs:NBBeams
value: int = driver.configure.signaling.nradio.cell.beams.nbBeams.get(cell_name␣
↪= 'abc')
```

Creates a <Number> of NZP CSI-RS beams within the active beam.

param cell_name
No help available

return
number: No help available

set(*cell_name: str*, *number: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BEAMs:NBBeams
driver.configure.signaling.nradio.cell.beams.nbBeams.set(cell_name = 'abc',␣
↪number = 1)
```

Creates a <Number> of NZP CSI-RS beams within the active beam.

param cell_name
No help available

param number
No help available

6.3.4.10.2.24 Bler

class BlerCls

Bler commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bler.clone()
```

Subgroups

6.3.4.10.2.25 Downlink

class DownlinkCls

Downlink commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bler.downlink.clone()
```

Subgroups

6.3.4.10.2.26 Percent

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BLER:DL:PERCent
```

class PercentCls

Percent commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BLER:DL:PERCent
value: float = driver.configure.signaling.nradio.cell.bler.downlink.percent.
↪get(cell_name = 'abc')
```

Configures the rate of transport block errors to be inserted into the downlink data, in percent, for the initial BWP.

param cell_name
No help available

return
percent: No help available

set(*cell_name: str*, *percent: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BLER:DL:PERCent
driver.configure.signaling.nradio.cell.bler.downlink.percent.set(cell_name =
↪'abc', percent = 1.0)
```

Configures the rate of transport block errors to be inserted into the downlink data, in percent, for the initial BWP.

param cell_name
No help available

param percent
No help available

6.3.4.10.2.27 Thousandth

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BLER:DL:THOusandth
```

class ThousandthCls

Thousandth commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BLER:DL:THOusandth
value: int = driver.configure.signaling.nradio.cell.bler.downlink.thousandth.
↪get(cell_name = 'abc')
```

Configures the rate of transport block errors to be inserted into the downlink data, in 1/1000th %, for the initial BWP.

param cell_name
No help available

return
thousandth: No help available

set(*cell_name: str*, *thousandth: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BLER:DL:THOusandth
driver.configure.signaling.nradio.cell.bler.downlink.thousandth.set(cell_name =
↪'abc', thousandth = 1)
```

Configures the rate of transport block errors to be inserted into the downlink data, in 1/1000th %, for the initial BWP.

param cell_name
No help available

param thousandth
No help available

6.3.4.10.2.28 Bwp<BwParts>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.configure.signaling.nradio.cell.bwp.repcap_bwParts_get()
driver.configure.signaling.nradio.cell.bwp.repcap_bwParts_set(repcap.BwParts.Nr1)
```

class BwpCls

Bwp commands group definition. 186 total commands, 19 Subgroups, 0 group commands Repeated Capability: BwParts, default value after init: BwParts.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.clone()
```

Subgroups

6.3.4.10.2.29 AsMode

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwpid>:ASMode
```

class AsModeCls

AsMode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → Asn1SignalMode

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwpid>:ASMode
value: enums.Asn1SignalMode = driver.configure.signaling.nradio.cell.bwp.asMode.
↪get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Selects the maximum number of BWPs signaled to the UE.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
asn_1_signal_mode: Based on UE capability, maximum 1 / 2 / 4 BWPs

set(*cell_name: str*, *asn_1_signal_mode: Asn1SignalMode*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwpid>:ASMode
driver.configure.signaling.nradio.cell.bwp.asMode.set(cell_name = 'abc', asn_1_
↪signal_mode = enums.Asn1SignalMode.B1, bwParts = repcap.BwParts.Default)
```

Selects the maximum number of BWPs signaled to the UE.

param cell_name
No help available

param asn_1_signal_mode
Based on UE capability, maximum 1 / 2 / 4 BWPs

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.30 Bler

class BlerCls

Bler commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.bler.clone()
```

Subgroups

6.3.4.10.2.31 Downlink

class DownlinkCls

Downlink commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.bler.downlink.clone()
```

Subgroups

6.3.4.10.2.32 Percent

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:BLER:DL:PERCent
```

class PercentCls

Percent commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → float

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:BLER:DL:PERCent
value: float = driver.configure.signaling.nradio.cell.bwp.bler.downlink.percent.
↪get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Configures the rate of transport block errors to be inserted into the downlink data, in percent, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
percent: No help available

set(*cell_name: str*, *percent: float*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:BLER:DL:PERCent
driver.configure.signaling.nradio.cell.bwp.bler.downlink.percent.set(cell_name␣
↪= 'abc', percent = 1.0, bwParts = repcap.BwParts.Default)
```

Configures the rate of transport block errors to be inserted into the downlink data, in percent, for BWP <bb>.

param cell_name
No help available

param percent
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.33 Thousandth

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:BLER:DL:THOusandth
```

class ThousandthCls

Thousandth commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:BLER:DL:THOusandth
value: int = driver.configure.signaling.nradio.cell.bwp.bler.downlink.
↪thousandth.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Configures the rate of transport block errors to be inserted into the downlink data, in 1/1000th %, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
thousandth: No help available

set(*cell_name: str*, *thousandth: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:BLER:DL:THOusandth
driver.configure.signaling.nradio.cell.bwp.bler.downlink.thousandth.set(cell_
↪name = 'abc', thousandth = 1, bwParts = repcap.BwParts.Default)
```

Configures the rate of transport block errors to be inserted into the downlink data, in 1/1000th %, for BWP <bb>.

param cell_name
No help available

param thousandth
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.34 CqiReporting

class CqiReportingCls

CqiReporting commands group definition. 10 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.cqiReporting.clone()
```

Subgroups

6.3.4.10.2.35 Combined

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:CQIReporting:COMBined
```

class CombinedCls

Combined commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Enable: enums.EnableCqi: No parameter help available
- Periodicity: enums.PeriodicityCqiReport: No parameter help available
- Offset: int: No parameter help available
- Ports: enums.Ports: No parameter help available
- Symbol: int: No parameter help available
- Power: enums.RsrcPower: No parameter help available
- Report_Offset: int: No parameter help available
- Report_Cqi: enums.ReportCqi: No parameter help available
- Report_Pmi: enums.ReportCqi: No parameter help available

class SetStruct

Structure for setting input parameters. Contains optional setting parameters. Fields:

- Cell_Name: str: No parameter help available
- Enable: enums.EnableCqi: No parameter help available
- Periodicity: enums.PeriodicityCqiReport: No parameter help available
- Offset: int: No parameter help available
- Ports: enums.Ports: No parameter help available
- Symbol: int: No parameter help available
- Power: enums.RsrcPower: No parameter help available
- Report_Offset: int: No parameter help available
- Report_Cqi: enums.ReportCqi: No parameter help available
- Report_Pmi: enums.ReportCqi: No parameter help available

get(*cell_name: str*, *bwParts=BwParts.Default*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:CQIReporting:COMBined
value: GetStruct = driver.configure.signaling.nradio.cell.bwp.cqiReporting.
↪combined.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

No command help available

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*structure: SetStruct*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:CQIReporting:COMBined
structure = driver.configure.signaling.nradio.cell.bwp.cqiReporting.combined.
↪SetStruct()
structure.Cell_Name: str = 'abc'
structure.Enable: enums.EnableCqi = enums.EnableCqi.APERiodic
structure.Periodicity: enums.PeriodicityCqiReport = enums.PeriodicityCqiReport.
↪P10
structure.Offset: int = 1
structure.Ports: enums.Ports = enums.Ports.P1
structure.Symbol: int = 1
structure.Power: enums.RsrcPower = enums.RsrcPower.M3DB
structure.Report_Offset: int = 1
structure.Report_Cqi: enums.ReportCqi = enums.ReportCqi.OFF
structure.Report_Pmi: enums.ReportCqi = enums.ReportCqi.OFF
driver.configure.signaling.nradio.cell.bwp.cqiReporting.combined.set(structure,␣
↪bwParts = repcap.BwParts.Default)
```

No command help available

param structure
for set value, see the help for SetStruct structure arguments.

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.36 Enable

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:CQIReporting:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → EnableCqi

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:CQIReporting:ENABle
value: enums.EnableCqi = driver.configure.signaling.nradio.cell.bwp.
↪cqiReporting.enable.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

No command help available

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
enable: No help available

set(*cell_name: str*, *enable: EnableCqi*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:CQIReporting:ENABle
driver.configure.signaling.nradio.cell.bwp.cqiReporting.enable.set(cell_name =
↪'abc', enable = enums.EnableCqi.APERiodic, bwParts = repcap.BwParts.Default)
```

No command help available

param cell_name
No help available

param enable
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.37 Periodicity

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:CQIReporting:PERiodicity
```

class PeriodicityCls

Periodicity commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → PeriodicityCqiReport

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:CQIReporting:PERiodicity
value: enums.PeriodicityCqiReport = driver.configure.signaling.nradio.cell.bwp.
↪cqiReporting.periodicity.get(cell_name = 'abc', bwParts = repcap.BwParts.
↪Default)
```

No command help available

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
periodicity: No help available

set(*cell_name: str*, *periodicity: PeriodicityCqiReport*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:CQIReporting:PERiodicity
driver.configure.signaling.nradio.cell.bwp.cqiReporting.periodicity.set(cell_
↪name = 'abc', periodicity = enums.PeriodicityCqiReport.P10, bwParts = repcap.
↪BwParts.Default)
```

No command help available

param cell_name
No help available

param periodicity
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.38 Report

class ReportCls

Report commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.cqiReporting.report.clone()
```

Subgroups

6.3.4.10.2.39 Cqi

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:CQIReporting:REPort:CQI
```

class CqiCls

Cqi commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → ReportCqi

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:CQIReporting:REPort:CQI
value: enums.ReportCqi = driver.configure.signaling.nradio.cell.bwp.
↪cqiReporting.report.cqi.get(cell_name = 'abc', bwParts = repcap.BwParts.
↪Default)
```

No command help available

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
report_format: No help available

set(*cell_name: str*, *report_format: ReportCqi*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:CQIReporting:REPort:CQI
driver.configure.signaling.nradio.cell.bwp.cqiReporting.report.cqi.set(cell_
↪name = 'abc', report_format = enums.ReportCqi.OFF, bwParts = repcap.BwParts.
↪Default)
```

No command help available

param cell_name
No help available

param report_format
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.40 Offset

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:CQIReporting:REPort:OFFSet
```

`class OffsetCls`

Offset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

`get`(*cell_name: str*, *bwParts=BwParts.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:CQIReporting:REPort:OFFSet
value: int = driver.configure.signaling.nradio.cell.bwp.cqiReporting.report.
↪offset.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

No command help available

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
offset: No help available

`set`(*cell_name: str*, *offset: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:CQIReporting:REPort:OFFSet
driver.configure.signaling.nradio.cell.bwp.cqiReporting.report.offset.set(cell_
↪name = 'abc', offset = 1, bwParts = repcap.BwParts.Default)
```

No command help available

param cell_name
No help available

param offset
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.41 Pmi

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:CQIReporting:REPort:PMI
```

class PmiCls

Pmi commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → ReportCqi

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:CQIReporting:REPort:PMI
value: enums.ReportCqi = driver.configure.signaling.nradio.cell.bwp.
↪cqiReporting.report.pmi.get(cell_name = 'abc', bwParts = repcap.BwParts.
↪Default)
```

No command help available

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
report_format: No help available

set(*cell_name: str*, *report_format: ReportCqi*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:CQIReporting:REPort:PMI
driver.configure.signaling.nradio.cell.bwp.cqiReporting.report.pmi.set(cell_
↪name = 'abc', report_format = enums.ReportCqi.OFF, bwParts = repcap.BwParts.
↪Default)
```

No command help available

param cell_name
No help available

param report_format
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.42 Resource

class ResourceCls

Resource commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.cqiReporting.resource.clone()
```

Subgroups

6.3.4.10.2.43 FoSymbol

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:CQIReporting:RESource:FOSYmbol
```

class FoSymbolCls

FoSymbol commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:CQIReporting:RESource:FOSYmbol
value: int = driver.configure.signaling.nradio.cell.bwp.cqiReporting.resource.
↪foSymbol.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

No command help available

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
symbol: No help available

set(*cell_name: str*, *symbol: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:CQIReporting:RESource:FOSYmbol
driver.configure.signaling.nradio.cell.bwp.cqiReporting.resource.foSymbol.
↪set(cell_name = 'abc', symbol = 1, bwParts = repcap.BwParts.Default)
```

No command help available

param cell_name
No help available

param symbol
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.44 Offset

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:CQIReporting:RESource:OFFSet
```

class OffsetCls

Offset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:CQIReporting:RESource:OFFSet
value: int = driver.configure.signaling.nradio.cell.bwp.cqiReporting.resource.
↪offset.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

No command help available

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
offset: No help available

set(*cell_name: str*, *offset: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:CQIReporting:RESource:OFFSet
driver.configure.signaling.nradio.cell.bwp.cqiReporting.resource.offset.
↪set(cell_name = 'abc', offset = 1, bwParts = repcap.BwParts.Default)
```

No command help available

param cell_name
No help available

param offset
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.45 Ports

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:CQIReporting:RESource:PORTs
```

class PortsCls

Ports commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → Ports

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:CQIReporting:RESource:PORTs
value: enums.Ports = driver.configure.signaling.nradio.cell.bwp.cqiReporting.
↪resource.ports.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

No command help available

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
ports: No help available

set(*cell_name: str*, *ports: Ports*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:CQIReporting:RESource:PORTs
driver.configure.signaling.nradio.cell.bwp.cqiReporting.resource.ports.set(cell_
↪name = 'abc', ports = enums.Ports.P1, bwParts = repcap.BwParts.Default)
```

No command help available

param cell_name
No help available

param ports
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.46 PoVsss

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:CQIReporting:RESource:POVSss
```

class PoVsssCls

PoVsss commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → RsrcPower

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:CQIReporting:RESource:POVSss
value: enums.RsrcPower = driver.configure.signaling.nradio.cell.bwp.
↪cqiReporting.resource.poVsss.get(cell_name = 'abc', bwParts = repcap.BwParts.
↪Default)
```

No command help available

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
power: No help available

set(*cell_name: str*, *power: RsrcPower*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:CQIReporting:RESource:POVSss
driver.configure.signaling.nradio.cell.bwp.cqiReporting.resource.poVsss.
↪set(cell_name = 'abc', power = enums.RsrcPower.M3DB, bwParts = repcap.BwParts.
↪Default)
```

No command help available

param cell_name
No help available

param power
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.47 Csi

class CsiCls

Csi commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.csi.clone()
```

Subgroups

6.3.4.10.2.48 Trs

class TrsCls

Trs commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.csi.trs.clone()
```

Subgroups

6.3.4.10.2.49 Config

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:CSI:TRS:CONFig
```

class ConfigCls

Config commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Index: int: Number of the TRS configuration.
- Bw_Selection: enums.BwSelection: All RBs of the BWP or maximum 52 RBs.
- Slot_Offset: int: Time domain offset.
- Symbol_Pair: enums.SymbolPair: Selects the two OFDM symbols used for TRS. The first digit indicates the first symbol. The remaining digits indicate the second symbol. Example: S913 means symbol 9 and symbol 13.
- Periodicity: enums.TrsPeriodicity: Periodicity for transmission of the resource set, in ms.
- No_Consec_Slots: int: Number of slots per resource set in FR2.

class SetStruct

Structure for setting input parameters. Fields:

- Cell_Name: str: No parameter help available
- Index: int: Number of the TRS configuration.
- Bw_Selection: enums.BwSelection: All RBs of the BWP or maximum 52 RBs.
- Slot_Offset: int: Time domain offset.
- Symbol_Pair: enums.SymbolPair: Selects the two OFDM symbols used for TRS. The first digit indicates the first symbol. The remaining digits indicate the second symbol. Example: S913 means symbol 9 and symbol 13.
- Periodicity: enums.TrsPperiodicity: Periodicity for transmission of the resource set, in ms.
- No_Consec_Slots: int: Number of slots per resource set in FR2.

get(*cell_name: str*, *bwParts=BwParts.Default*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:CSI:TRS:CONFig
value: GetStruct = driver.configure.signaling.nradio.cell.bwp.csi.trs.config.
↪get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Defines settings of TRS <Index>, for BWP <bb>. If there are several TRS configurations, a query returns the settings of all TRS configurations.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*structure: SetStruct*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:CSI:TRS:CONFig
structure = driver.configure.signaling.nradio.cell.bwp.csi.trs.config.
↪SetStruct()
structure.Cell_Name: str = 'abc'
structure.Index: int = 1
structure.Bw_Selection: enums.BwSelection = enums.BwSelection.ALL
structure.Slot_Offset: int = 1
structure.Symbol_Pair: enums.SymbolPair = enums.SymbolPair.S04
structure.Periodicity: enums.TrsPeriodicity = enums.TrsPeriodicity.P10
structure.No_Consec_Slots: int = 1
driver.configure.signaling.nradio.cell.bwp.csi.trs.config.set(structure,␣
↪bwParts = repcap.BwParts.Default)
```

Defines settings of TRS <Index>, for BWP <bb>. If there are several TRS configurations, a query returns the settings of all TRS configurations.

param structure
for set value, see the help for SetStruct structure arguments.

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.50 Mode

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:CSI:TRS:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → ModeTrs

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:CSI:TRS:MODE
value: enums.ModeTrs = driver.configure.signaling.nradio.cell.bwp.csi.trs.mode.
↪get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Selects the configuration mode for TRS transmission, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
mode: OFF: no TRS DEF: TRS according to 3GPP TS 38.508 UDEF: user-defined TRS

set(*cell_name: str*, *mode: ModeTrs*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:CSI:TRS:MODE
driver.configure.signaling.nradio.cell.bwp.csi.trs.mode.set(cell_name = 'abc',␣
↪mode = enums.ModeTrs.DEF, bwParts = repcap.BwParts.Default)
```

Selects the configuration mode for TRS transmission, for BWP <bb>.

param cell_name
No help available

param mode
OFF: no TRS DEF: TRS according to 3GPP TS 38.508 UDEF: user-defined TRS

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.51 Dmrs

class DmrsCls

Dmrs commands group definition. 21 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.dmrs.clone()
```

Subgroups

6.3.4.10.2.52 Downlink

class DownlinkCls

Downlink commands group definition. 10 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.dmrs.downlink.clone()
```

Subgroups

6.3.4.10.2.53 Mta

class MtaCls

Mta commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.dmrs.downlink.mta.clone()
```

Subgroups

6.3.4.10.2.54 Length

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:DL:MTA:LENGth
```

class LengthCls

Length commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → MaxLength

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:DL:MTA:LENGth
value: enums.MaxLength = driver.configure.signaling.nradio.cell.bwp.dmrs.
↪downlink.mta.length.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

No command help available

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
max_length: No help available

set(*cell_name: str*, *max_length: MaxLength*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:DL:MTA:LENGth
driver.configure.signaling.nradio.cell.bwp.dmrs.downlink.mta.length.set(cell_
↪name = 'abc', max_length = enums.MaxLength.L1, bwParts = repcap.BwParts.
↪Default)
```

No command help available

param cell_name
No help available

param max_length
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.55 Papr

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:DL:MTA:PAPR
```

class PaprCls

Papr commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:DL:MTA:PAPR
value: bool = driver.configure.signaling.nradio.cell.bwp.dmrs.downlink.mta.papr.
↪get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Enables the usage of a DMRS with a low PAPR, for PDSCH, mapping type A, BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
enable: No help available

set(*cell_name: str*, *enable: bool*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:DL:MTA:PAPR
driver.configure.signaling.nradio.cell.bwp.dmrs.downlink.mta.papr.set(cell_name␣
↪= 'abc', enable = False, bwParts = repcap.BwParts.Default)
```

Enables the usage of a DMRS with a low PAPR, for PDSCH, mapping type A, BWP <bb>.

param cell_name
No help available

param enable
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.56 Position

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:DL:MTA:POSition
```

class PositionCls

Position commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → MtxPosition

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:DL:MTA:POSition
value: enums.MtxPosition = driver.configure.signaling.nradio.cell.bwp.dmrs.
↪downlink.mta.position.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Defines parameter 'dmrs-AdditionalPosition' for PDSCH, mapping type A, BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
position: No help available

set(*cell_name: str*, *position: MtxPosition*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:DL:MTA:POSition
driver.configure.signaling.nradio.cell.bwp.dmrs.downlink.mta.position.set(cell_
```

(continues on next page)

(continued from previous page)

```
↪name = 'abc', position = enums.MtxPosition.P0, bwParts = repcap.BwParts.
↪Default)
```

Defines parameter 'dmrs-AdditionalPosition' for PDSCH, mapping type A, BWP <bb>.

param cell_name
No help available

param position
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.57 TypePy

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:DL:MTA:TYPE
```

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → ConfigType

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:DL:MTA:TYPE
value: enums.ConfigType = driver.configure.signaling.nradio.cell.bwp.dmrs.
↪downlink.mta.typePy.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

No command help available

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
config_type: No help available

set(*cell_name: str*, *config_type: ConfigType*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:DL:MTA:TYPE
driver.configure.signaling.nradio.cell.bwp.dmrs.downlink.mta.typePy.set(cell_
↪name = 'abc', config_type = enums.ConfigType.T1, bwParts = repcap.BwParts.
↪Default)
```

No command help available

param cell_name
No help available

param config_type
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.58 Mtb

class MtbCls

Mtb commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.dmrs.downlink.mtb.clone()
```

Subgroups

6.3.4.10.2.59 Length

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:DL:MTB:LENGth
```

class LengthCls

Length commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → MaxLength

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:DL:MTB:LENGth
value: enums.MaxLength = driver.configure.signaling.nradio.cell.bwp.dmrs.
↪downlink.mtb.length.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

No command help available

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
max_length: No help available

set(*cell_name: str*, *max_length: MaxLength*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:DL:MTB:LENGth
driver.configure.signaling.nradio.cell.bwp.dmrs.downlink.mtb.length.set(cell_
↪name = 'abc', max_length = enums.MaxLength.L1, bwParts = repcap.BwParts.
↪Default)
```

No command help available

param cell_name
No help available

param max_length
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.60 Papr

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:DL:MTB:PAPR
```

class PaprCls

Papr commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:DL:MTB:PAPR
value: bool = driver.configure.signaling.nradio.cell.bwp.dmrs.downlink.mtb.papr.
↪get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Enables the usage of a DMRS with a low PAPR, for PDSCH, mapping type B, BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
enable: No help available

set(*cell_name: str*, *enable: bool*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:DL:MTB:PAPR
driver.configure.signaling.nradio.cell.bwp.dmrs.downlink.mtb.papr.set(cell_name␣
↪= 'abc', enable = False, bwParts = repcap.BwParts.Default)
```

Enables the usage of a DMRS with a low PAPR, for PDSCH, mapping type B, BWP <bb>.

param cell_name
No help available

param enable
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.61 Position

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:DL:MTB:POSition
```

class PositionCls

Position commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → MtxPosition

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:DL:MTB:POSition
value: enums.MtxPosition = driver.configure.signaling.nradio.cell.bwp.dmrs.
↪downlink.mtb.position.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Defines parameter 'dmrs-AdditionalPosition' for PDSCH, mapping type B, BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
position: No help available

set(*cell_name: str*, *position: MtxPosition*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:DL:MTB:POSition
driver.configure.signaling.nradio.cell.bwp.dmrs.downlink.mtb.position.set(cell_
↪name = 'abc', position = enums.MtxPosition.P0, bwParts = repcap.BwParts.
↪Default)
```

Defines parameter 'dmrs-AdditionalPosition' for PDSCH, mapping type B, BWP <bb>.

param cell_name
No help available

param position
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.62 TypePy

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:DL:MTB:TYPE
```

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → ConfigType

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:DL:MTB:TYPE
value: enums.ConfigType = driver.configure.signaling.nradio.cell.bwp.dmrs.
↪downlink.mtb.typePy.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

No command help available

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
config_type: No help available

set(*cell_name: str*, *config_type: ConfigType*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:DL:MTB:TYPE
driver.configure.signaling.nradio.cell.bwp.dmrs.downlink.mtb.typePy.set(cell_
↪name = 'abc', config_type = enums.ConfigType.T1, bwParts = repcap.BwParts.
↪Default)
```

No command help available

param cell_name
No help available

param config_type
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.63 Ptrs

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:DL:PTRS
```

class PtrsCls

Ptrs commands group definition. 2 total commands, 1 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Time_Dens_Pres: enums.DensityPreset: Presence of the 'timeDensity' (field signaled or not) .
- Time_Dens_Mcs_1: int: Signaled 'timeDensity', ptrs-MCS1.
- Time_Dens_Mcs_2: int: Signaled 'timeDensity', ptrs-MCS2.
- Time_Dens_Mcs_3: int: Signaled 'timeDensity', ptrs-MCS3.
- Freq_Dens_Pres: enums.DensityPreset: Presence of the 'frequencyDensity' (field signaled or not) .

- Freq_Dens_Nrb_0: int: Signaled 'frequencyDensity', NRB0.
- Freq_Dens_Nrb_1: int: Signaled 'frequencyDensity', NRB1.
- Epre_Ratio: enums.EpreRatio: Signaled 'epre-Ratio', PTRS relative to PDSCH.
- Resource_Offset: enums.ResourceOffset: Signaled 'resourceElementOffset'.

class SetStruct

Structure for setting input parameters. Fields:

- Cell_Name: str: No parameter help available
- Time_Dens_Pres: enums.DensityPreset: Presence of the 'timeDensity' (field signaled or not) .
- Time_Dens_Mcs_1: int: Signaled 'timeDensity', ptrs-MCS1.
- Time_Dens_Mcs_2: int: Signaled 'timeDensity', ptrs-MCS2.
- Time_Dens_Mcs_3: int: Signaled 'timeDensity', ptrs-MCS3.
- Freq_Dens_Pres: enums.DensityPreset: Presence of the 'frequencyDensity' (field signaled or not) .
- Freq_Dens_Nrb_0: int: Signaled 'frequencyDensity', NRB0.
- Freq_Dens_Nrb_1: int: Signaled 'frequencyDensity', NRB1.
- Epre_Ratio: enums.EpreRatio: Signaled 'epre-Ratio', PTRS relative to PDSCH.
- Resource_Offset: enums.ResourceOffset: Signaled 'resourceElementOffset'.

get(*cell_name: str*, *bwParts=BwParts.Default*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:DL:PTRS
value: GetStruct = driver.configure.signaling.nradio.cell.bwp.dmrs.downlink.
↪ptrs.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Defines the IE 'PTRS-DownlinkConfig' for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*structure: SetStruct*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:DL:PTRS
structure = driver.configure.signaling.nradio.cell.bwp.dmrs.downlink.ptrs.
↪SetStruct()
structure.Cell_Name: str = 'abc'
structure.Time_Dens_Pres: enums.DensityPreset = enums.DensityPreset.NPResent
structure.Time_Dens_Mcs_1: int = 1
structure.Time_Dens_Mcs_2: int = 1
structure.Time_Dens_Mcs_3: int = 1
structure.Freq_Dens_Pres: enums.DensityPreset = enums.DensityPreset.NPResent
structure.Freq_Dens_Nrb_0: int = 1
structure.Freq_Dens_Nrb_1: int = 1
```

(continues on next page)

(continued from previous page)

```
structure.Epre_Ratio: enums.EpreRatio = enums.EpreRatio.R0
structure.Resource_Offset: enums.ResourceOffset = enums.ResourceOffset.NPResent
driver.configure.signaling.nradio.cell.bwp.dmrs.downlink.ptrs.set(structure,␣
↪bwParts = repcap.BwParts.Default)
```

Defines the IE 'PTRS-DownlinkConfig' for BWP <bb>.

param structure
for set value, see the help for SetStruct structure arguments.

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.dmrs.downlink.ptrs.clone()
```

Subgroups

6.3.4.10.2.64 Enable

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:DL:PTRS:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:DL:PTRS:ENABle
value: bool = driver.configure.signaling.nradio.cell.bwp.dmrs.downlink.ptrs.
↪enable.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Enables sending the IE 'PTRS-DownlinkConfig' for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
enable: No help available

set(*cell_name: str*, *enable: bool*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:DL:PTRS:ENABle
driver.configure.signaling.nradio.cell.bwp.dmrs.downlink.ptrs.enable.set(cell_
↪name = 'abc', enable = False, bwParts = repcap.BwParts.Default)
```

Enables sending the IE 'PTRS-DownlinkConfig' for BWP <bb>.

param cell_name
No help available

param enable
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.65 Uplink

class UplinkCls

Uplink commands group definition. 11 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.dmrs.uplink.clone()
```

Subgroups

6.3.4.10.2.66 Mta

class MtaCls

Mta commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.dmrs.uplink.mta.clone()
```

Subgroups

6.3.4.10.2.67 Length

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:UL:MTA:LENGth
```

class LengthCls

Length commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → MaxLength

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:UL:MTA:LENGth
value: enums.MaxLength = driver.configure.signaling.nradio.cell.bwp.dmrs.uplink.
↪mta.length.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

No command help available

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
max_length: No help available

set(*cell_name: str*, *max_length: MaxLength*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:UL:MTA:LENGth
driver.configure.signaling.nradio.cell.bwp.dmrs.uplink.mta.length.set(cell_name␣
↪= 'abc', max_length = enums.MaxLength.L1, bwParts = repcap.BwParts.Default)
```

No command help available

param cell_name
No help available

param max_length
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.68 Papr

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:UL:MTA:PAPR
```

class PaprCls

Papr commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:UL:MTA:PAPR
value: bool = driver.configure.signaling.nradio.cell.bwp.dmrs.uplink.mta.papr.
↪get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Enables the usage of a DMRS with a low PAPR, for PUSCH, mapping type A, BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
enable: No help available

set(*cell_name: str*, *enable: bool*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:UL:MTA:PAPR
driver.configure.signaling.nradio.cell.bwp.dmrs.uplink.mta.papr.set(cell_name =
↪'abc', enable = False, bwParts = repcap.BwParts.Default)
```

Enables the usage of a DMRS with a low PAPR, for PUSCH, mapping type A, BWP <bb>.

param cell_name
No help available

param enable
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.69 Position

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:UL:MTA:POSition
```

class PositionCls

Position commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → MtxPosition

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:UL:MTA:POSition
value: enums.MtxPosition = driver.configure.signaling.nradio.cell.bwp.dmrs.
↪uplink.mta.position.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Defines parameter 'dmrs-AdditionalPosition' for PUSCH, mapping type A, BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
position: No help available

set(*cell_name: str*, *position: MtxPosition*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:UL:MTA:POSition
driver.configure.signaling.nradio.cell.bwp.dmrs.uplink.mta.position.set(cell_
↪name = 'abc', position = enums.MtxPosition.P0, bwParts = repcap.BwParts.
↪Default)
```

Defines parameter 'dmrs-AdditionalPosition' for PUSCH, mapping type A, BWP <bb>.

param cell_name
No help available

param position
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.70 TypePy

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:UL:MTA:TYPE
```

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → ConfigType

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:UL:MTA:TYPE
value: enums.ConfigType = driver.configure.signaling.nradio.cell.bwp.dmrs.
↪uplink.mta.typePy.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

No command help available

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
config_type: No help available

set(*cell_name: str*, *config_type: ConfigType*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:UL:MTA:TYPE
driver.configure.signaling.nradio.cell.bwp.dmrs.uplink.mta.typePy.set(cell_name␣
↪= 'abc', config_type = enums.ConfigType.T1, bwParts = repcap.BwParts.Default)
```

No command help available

param cell_name
No help available

param config_type
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.71 Mtb

class MtbCls

Mtb commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.dmrs.uplink.mtb.clone()
```

Subgroups

6.3.4.10.2.72 Length

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:UL:MTB:LENGth
```

class LengthCls

Length commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → MaxLength

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:UL:MTB:LENGth
value: enums.MaxLength = driver.configure.signaling.nradio.cell.bwp.dmrs.uplink.
↪mtb.length.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

No command help available

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
max_length: No help available

set(*cell_name: str*, *max_length: MaxLength*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:UL:MTB:LENGth
driver.configure.signaling.nradio.cell.bwp.dmrs.uplink.mtb.length.set(cell_name␣
↪= 'abc', max_length = enums.MaxLength.L1, bwParts = repcap.BwParts.Default)
```

No command help available

param cell_name
No help available

param max_length
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.73 Papr

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:UL:MTB:PAPR
```

class PaprCls

Papr commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:UL:MTB:PAPR
value: bool = driver.configure.signaling.nradio.cell.bwp.dmrs.uplink.mtb.papr.
↪get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Enables the usage of a DMRS with a low PAPR, for PUSCH, mapping type B, BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
enable: No help available

set(*cell_name: str*, *enable: bool*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:UL:MTB:PAPR
driver.configure.signaling.nradio.cell.bwp.dmrs.uplink.mtb.papr.set(cell_name =
↪'abc', enable = False, bwParts = repcap.BwParts.Default)
```

Enables the usage of a DMRS with a low PAPR, for PUSCH, mapping type B, BWP <bb>.

param cell_name
No help available

param enable
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.74 Position

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:UL:MTB:POSition
```

class PositionCls

Position commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → MtxPosition

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:UL:MTB:POSition
value: enums.MtxPosition = driver.configure.signaling.nradio.cell.bwp.dmrs.
↪uplink.mtb.position.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Defines parameter 'dmrs-AdditionalPosition' for PUSCH, mapping type B, BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
position: No help available

set(*cell_name: str*, *position: MtxPosition*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:UL:MTB:POSition
driver.configure.signaling.nradio.cell.bwp.dmrs.uplink.mtb.position.set(cell_
↪name = 'abc', position = enums.MtxPosition.P0, bwParts = repcap.BwParts.
↪Default)
```

Defines parameter 'dmrs-AdditionalPosition' for PUSCH, mapping type B, BWP <bb>.

param cell_name
No help available

param position
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.75 TypePy

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:UL:MTB:TYPE
```

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → ConfigType

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:UL:MTB:TYPE
value: enums.ConfigType = driver.configure.signaling.nradio.cell.bwp.dmrs.
↪uplink.mtb.typePy.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

No command help available

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
config_type: No help available

set(*cell_name: str*, *config_type: ConfigType*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:UL:MTB:TYPE
driver.configure.signaling.nradio.cell.bwp.dmrs.uplink.mtb.typePy.set(cell_name␣
↪= 'abc', config_type = enums.ConfigType.T1, bwParts = repcap.BwParts.Default)
```

No command help available

param cell_name
No help available

param config_type
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.76 Ptrs

class PtrsCls

Ptrs commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.dmrs.uplink.ptrs.clone()
```

Subgroups

6.3.4.10.2.77 Enable

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:UL:PTRS:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:UL:PTRS:ENABle
value: bool = driver.configure.signaling.nradio.cell.bwp.dmrs.uplink.ptrs.
↪enable.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Enables sending the IE 'PTRS-UplinkConfig' for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
enable: No help available

set(*cell_name: str*, *enable: bool*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:UL:PTRS:ENABle
driver.configure.signaling.nradio.cell.bwp.dmrs.uplink.ptrs.enable.set(cell_
↪name = 'abc', enable = False, bwParts = repcap.BwParts.Default)
```

Enables sending the IE 'PTRS-UplinkConfig' for BWP <bb>.

param cell_name
No help available

param enable
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.78 TpDisable

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:UL:PTRS:TPDisable
```

class TpDisableCls

TpDisable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Time_Dens_Pres: enums.DensityPreset: Presence of the 'timeDensity' (field signaled or not) .
- Time_Dens_Mcs_1: int: Signaled 'timeDensity', ptrs-MCS1.
- Time_Dens_Mcs_2: int: Signaled 'timeDensity', ptrs-MCS2.
- Time_Dens_Mcs_3: int: Signaled 'timeDensity', ptrs-MCS3.
- Freq_Dens_Pres: enums.DensityPreset: Presence of the 'frequencyDensity' (field signaled or not) .
- Freq_Dens_Nrb_0: int: Signaled 'frequencyDensity', NRB0.
- Freq_Dens_Nrb_1: int: Signaled 'frequencyDensity', NRB1.
- Max_Ports: enums.MaxPorts: Signaled 'maxNrofPorts'.
- Resource_Offset: enums.ResourceOffset: Signaled 'resourceElementOffset'.
- Power: enums.PtrsPower: Signaled 'ptrs-Power'.

class SetStruct

Structure for setting input parameters. Fields:

- Cell_Name: str: No parameter help available
- Time_Dens_Pres: enums.DensityPreset: Presence of the 'timeDensity' (field signaled or not) .
- Time_Dens_Mcs_1: int: Signaled 'timeDensity', ptrs-MCS1.
- Time_Dens_Mcs_2: int: Signaled 'timeDensity', ptrs-MCS2.
- Time_Dens_Mcs_3: int: Signaled 'timeDensity', ptrs-MCS3.
- Freq_Dens_Pres: enums.DensityPreset: Presence of the 'frequencyDensity' (field signaled or not) .
- Freq_Dens_Nrb_0: int: Signaled 'frequencyDensity', NRB0.
- Freq_Dens_Nrb_1: int: Signaled 'frequencyDensity', NRB1.
- Max_Ports: enums.MaxPorts: Signaled 'maxNrofPorts'.
- Resource_Offset: enums.ResourceOffset: Signaled 'resourceElementOffset'.
- Power: enums.PtrsPower: Signaled 'ptrs-Power'.

get(*cell_name: str*, *bwParts=BwParts.Default*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:UL:PTRS:TPDisable
value: GetStruct = driver.configure.signaling.nradio.cell.bwp.dmrs.uplink.ptrs.
↪tpDisable.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Defines the IE 'PTRS-UplinkConfig' for signals without transform precoding, BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*structure: SetStruct*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:UL:PTRS:TPDisable
structure = driver.configure.signaling.nradio.cell.bwp.dmrs.uplink.ptrs.
↪tpDisable.SetStruct()
structure.Cell_Name: str = 'abc'
structure.Time_Dens_Pres: enums.DensityPreset = enums.DensityPreset.NPResent
structure.Time_Dens_Mcs_1: int = 1
structure.Time_Dens_Mcs_2: int = 1
structure.Time_Dens_Mcs_3: int = 1
structure.Freq_Dens_Pres: enums.DensityPreset = enums.DensityPreset.NPResent
structure.Freq_Dens_Nrb_0: int = 1
structure.Freq_Dens_Nrb_1: int = 1
structure.Max_Ports: enums.MaxPorts = enums.MaxPorts.N1
structure.Resource_Offset: enums.ResourceOffset = enums.ResourceOffset.NPResent
structure.Power: enums.PtrsPower = enums.PtrsPower.P00
driver.configure.signaling.nradio.cell.bwp.dmrs.uplink.ptrs.tpDisable.
↪set(structure, bwParts = repcap.BwParts.Default)
```

Defines the IE 'PTRS-UplinkConfig' for signals without transform precoding, BWP <bb>.

param structure
for set value, see the help for SetStruct structure arguments.

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.79 TpEnable

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:UL:PTRS:TPENable
```

class TpEnableCls

TpEnable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Sample_Dens_Nrb_0: int: Signaled 'sampleDensity', NRB0.
- Sample_Dens_Nrb_1: int: Signaled 'sampleDensity', NRB1.
- Sample_Dens_Nrb_2: int: Signaled 'sampleDensity', NRB2.
- Sample_Dens_Nrb_3: int: Signaled 'sampleDensity', NRB3.

- Sample_Dens_Nrb_4: int: Signaled 'sampleDensity', NRB4.
- Tp_Time_Dens: enums.TpTimeDens: Signaled 'timeDensityTransformPrecoding'.

class SetStruct

Structure for setting input parameters. Fields:

- Cell_Name: str: No parameter help available
- Sample_Dens_Nrb_0: int: Signaled 'sampleDensity', NRB0.
- Sample_Dens_Nrb_1: int: Signaled 'sampleDensity', NRB1.
- Sample_Dens_Nrb_2: int: Signaled 'sampleDensity', NRB2.
- Sample_Dens_Nrb_3: int: Signaled 'sampleDensity', NRB3.
- Sample_Dens_Nrb_4: int: Signaled 'sampleDensity', NRB4.
- Tp_Time_Dens: enums.TpTimeDens: Signaled 'timeDensityTransformPrecoding'.

get(*cell_name: str*, *bwParts=BwParts.Default*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:UL:PTRS:TPENable
value: GetStruct = driver.configure.signaling.nradio.cell.bwp.dmrs.uplink.ptrs.
↪tpEnable.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Defines the IE 'PTRS-UplinkConfig' for signals with transform precoding, BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*structure: SetStruct*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:DMRS:UL:PTRS:TPENable
structure = driver.configure.signaling.nradio.cell.bwp.dmrs.uplink.ptrs.
↪tpEnable.SetStruct()
structure.Cell_Name: str = 'abc'
structure.Sample_Dens_Nrb_0: int = 1
structure.Sample_Dens_Nrb_1: int = 1
structure.Sample_Dens_Nrb_2: int = 1
structure.Sample_Dens_Nrb_3: int = 1
structure.Sample_Dens_Nrb_4: int = 1
structure.Tp_Time_Dens: enums.TpTimeDens = enums.TpTimeDens.D2
driver.configure.signaling.nradio.cell.bwp.dmrs.uplink.ptrs.tpEnable.
↪set(structure, bwParts = repcap.BwParts.Default)
```

Defines the IE 'PTRS-UplinkConfig' for signals with transform precoding, BWP <bb>.

param structure
for set value, see the help for SetStruct structure arguments.

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.80 Downlink

class DownlinkCls

Downlink commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.downlink.clone()
```

Subgroups

6.3.4.10.2.81 Default

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwpid>:DL:DEFault
```

class DefaultCls

Default commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwpid>:DL:DEFault
value: int = driver.configure.signaling.nradio.cell.bwp.downlink.default.
↪get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Selects the default BWP for the DL, via its ID.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
default_dl_bwp: No help available

set(*cell_name: str*, *default_dl_bwp: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwpid>:DL:DEFault
driver.configure.signaling.nradio.cell.bwp.downlink.default.set(cell_name = 'abc
↪', default_dl_bwp = 1, bwParts = repcap.BwParts.Default)
```

Selects the default BWP for the DL, via its ID.

param cell_name
No help available

param default_dl_bwp
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.82 LbWidth

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwpid>:DL:LBWidth
```

class LbWidthCls

LbWidth commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwpid>:DL:LBWidth
value: int = driver.configure.signaling.nradio.cell.bwp.downlink.lbWidth.
↪get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Defines the resource indication value (RIV) signaled as 'locationAndBandwidth', for the downlink, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
riv: No help available

set(*cell_name: str*, *riv: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwpid>:DL:LBWidth
driver.configure.signaling.nradio.cell.bwp.downlink.lbWidth.set(cell_name = 'abc
↪', riv = 1, bwParts = repcap.BwParts.Default)
```

Defines the resource indication value (RIV) signaled as 'locationAndBandwidth', for the downlink, for BWP <bb>.

param cell_name
No help available

param riv
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.83 Rb

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwpid>:DL:RB
```

class RbCls

Rb commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Number_Rb: int: No parameter help available
- Start_Rb: int: No parameter help available

get(*cell_name: str*, *bwParts=BwParts.Default*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwpid>:DL:RB
value: GetStruct = driver.configure.signaling.nradio.cell.bwp.downlink.rb.
↪get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Defines the downlink of BWP <bb> in the frequency domain as a contiguous set of RBs, within the carrier bandwidth.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *number_rb: int*, *start_rb: int = None*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwpid>:DL:RB
driver.configure.signaling.nradio.cell.bwp.downlink.rb.set(cell_name = 'abc',␣
↪number_rb = 1, start_rb = 1, bwParts = repcap.BwParts.Default)
```

Defines the downlink of BWP <bb> in the frequency domain as a contiguous set of RBs, within the carrier bandwidth.

param cell_name
No help available

param number_rb
No help available

param start_rb
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.84 Harq

class HarqCls

Harq commands group definition. 18 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.harq.clone()
```

Subgroups

6.3.4.10.2.85 Downlink

class DownlinkCls

Downlink commands group definition. 9 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.harq.downlink.clone()
```

Subgroups

6.3.4.10.2.86 Auto

class AutoCls

Auto commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.harq.downlink.auto.clone()
```

Subgroups

6.3.4.10.2.87 Mret

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:DL:AUTO:MRET
```

class MretCls

Mret commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:DL:AUTO:MRET
value: int = driver.configure.signaling.nradio.cell.bwp.harq.downlink.auto.mret.
↪get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Configures the maximum number of retransmissions, for auto-configured DL HARQ, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
retransmissions: No help available

set(*cell_name: str*, *retransmissions: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:DL:AUTO:MRET
driver.configure.signaling.nradio.cell.bwp.harq.downlink.auto.mret.set(cell_
↪name = 'abc', retransmissions = 1, bwParts = repcap.BwParts.Default)
```

Configures the maximum number of retransmissions, for auto-configured DL HARQ, for BWP <bb>.

param cell_name
No help available

param retransmissions
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.88 Cmode

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:DL:CMODe
```

class CmodeCls

Cmode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → ModeC

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:DL:CMODe
value: enums.ModeC = driver.configure.signaling.nradio.cell.bwp.harq.downlink.
↪cmode.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Selects a mode for DL HARQ configuration, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
mode: NOTC: no DL HARQ AUTO: automatic configuration of the DL HARQ settings USER: user-defined configuration of the DL HARQ settings

set(*cell_name: str*, *mode: ModeC*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:DL:CMODe
driver.configure.signaling.nradio.cell.bwp.harq.downlink.cmode.set(cell_name =
↪'abc', mode = enums.ModeC.AUTO, bwParts = repcap.BwParts.Default)
```

Selects a mode for DL HARQ configuration, for BWP <bb>.

param cell_name
No help available

param mode
NOTC: no DL HARQ AUTO: automatic configuration of the DL HARQ settings USER: user-defined configuration of the DL HARQ settings

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.89 User

class UserCls

User commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.harq.downlink.user.clone()
```

Subgroups

6.3.4.10.2.90 Ack

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:DL:USER:ACK
```

class AckCls

Ack commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → AckOrDtx

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:DL:USER:ACK
value: enums.AckOrDtx = driver.configure.signaling.nradio.cell.bwp.harq.
↪downlink.user.ack.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Defines the reaction to ACKs sent by the UE, for user-defined DL HARQ, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
ack: STOP: stop retransmitting CONTinue: continue retransmitting

set(*cell_name: str*, *ack: AckOrDtx*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:DL:USER:ACK
driver.configure.signaling.nradio.cell.bwp.harq.downlink.user.ack.set(cell_name␣
↪= 'abc', ack = enums.AckOrDtx.CONTinue, bwParts = repcap.BwParts.Default)
```

Defines the reaction to ACKs sent by the UE, for user-defined DL HARQ, for BWP <bb>.

param cell_name
No help available

param ack
STOP: stop retransmitting CONTinue: continue retransmitting

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.91 Dtx

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:DL:USER:DTX
```

class DtxCls

Dtx commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → AckOrDtx

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:DL:USER:DTX
value: enums.AckOrDtx = driver.configure.signaling.nradio.cell.bwp.harq.
↪downlink.user.dtx.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Defines the reaction to DTX (missing ACKs) , for user-defined DL HARQ, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
dtx: STOP: stop retransmitting CONTinue: continue retransmitting

set(*cell_name: str*, *dtx: AckOrDtx*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:DL:USER:DTX
driver.configure.signaling.nradio.cell.bwp.harq.downlink.user.dtx.set(cell_name␣
↪= 'abc', dtx = enums.AckOrDtx.CONTinue, bwParts = repcap.BwParts.Default)
```

Defines the reaction to DTX (missing ACKs) , for user-defined DL HARQ, for BWP <bb>.

param cell_name
No help available

param dtx
STOP: stop retransmitting CONTinue: continue retransmitting

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.92 MinOffset

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:DL:USER:MINoffset
```

class MinOffsetCls

MinOffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:DL:USER:MINoffset
value: int = driver.configure.signaling.nradio.cell.bwp.harq.downlink.user.
↪minOffset.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Defines the minimum offset for user-defined DL HARQ, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
offset: No help available

set(*cell_name: str*, *offset: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:DL:USER:MINoffset
driver.configure.signaling.nradio.cell.bwp.harq.downlink.user.minOffset.
↪set(cell_name = 'abc', offset = 1, bwParts = repcap.BwParts.Default)
```

Defines the minimum offset for user-defined DL HARQ, for BWP <bb>.

param cell_name
No help available

param offset
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.93 Retransm

class RetransmCls

Retransm commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.harq.downlink.user.retransm.clone()
```

Subgroups

6.3.4.10.2.94 Ariv

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:DL:USER:RETRansm:ARIV
```

class ArivCls

Ariv commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *index: int*, *bwParts=BwParts.Default*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:DL:USER:RETRansm:ARIV
value: bool = driver.configure.signaling.nradio.cell.bwp.harq.downlink.user.
↪retransm.ariv.get(cell_name = 'abc', index = 1, bwParts = repcap.BwParts.
↪Default)
```

Configures auto RIV for a certain retransmission, for user-defined DL HARQ, for BWP <bb>.

param cell_name
No help available

param index
Index of the retransmission

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
riv: ON: Auto RIV enabled, no. of RB and start RB set automatically. OFF: Auto RIV disabled, you can define no. of RB and start RB.

set(*cell_name: str*, *index: int*, *riv: bool*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:DL:USER:RETRansm:ARIV
driver.configure.signaling.nradio.cell.bwp.harq.downlink.user.retransm.ariv.
↪set(cell_name = 'abc', index = 1, riv = False, bwParts = repcap.BwParts.
↪Default)
```

Configures auto RIV for a certain retransmission, for user-defined DL HARQ, for BWP <bb>.

param cell_name
No help available

param index
Index of the retransmission

param riv
ON: Auto RIV enabled, no. of RB and start RB set automatically. OFF: Auto RIV disabled, you can define no. of RB and start RB.

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.95 Modulation

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:DL:USER:RETRansm:MODulation
```

class ModulationCls

Modulation commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *index: int*, *bwParts=BwParts.Default*) → ModulationRetr

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:HARQ:DL:USER:RETRansm:MODulation
value: enums.ModulationRetr = driver.configure.signaling.nradio.cell.bwp.harq.
↪downlink.user.retransm.modulation.get(cell_name = 'abc', index = 1, bwParts =␣
↪repcap.BwParts.Default)
```

Selects a modulation scheme for a certain retransmission, for user-defined DL HARQ, for BWP <bb>.

param cell_name
No help available

param index
Index of the retransmission

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
modulation: BPSK, auto mode, /2-BPSK, QPSK, 16QAM, 64QAM, 256QAM, 1024QAM

set(*cell_name: str*, *index: int*, *modulation: ModulationRetr*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:HARQ:DL:USER:RETRansm:MODulation
driver.configure.signaling.nradio.cell.bwp.harq.downlink.user.retransm.
↪modulation.set(cell_name = 'abc', index = 1, modulation = enums.
↪ModulationRetr.AUTO, bwParts = repcap.BwParts.Default)
```

Selects a modulation scheme for a certain retransmission, for user-defined DL HARQ, for BWP <bb>.

param cell_name
No help available

param index
Index of the retransmission

param modulation
BPSK, auto mode, /2-BPSK, QPSK, 16QAM, 64QAM, 256QAM, 1024QAM

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.96 Rb

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:DL:USER:RETRansm:RB
```

class RbCls

Rb commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Nrb: int: No parameter help available
- Start_Rb: int: No parameter help available

get(*cell_name: str*, *index: int*, *bwParts=BwParts.Default*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:DL:USER:RETRansm:RB
value: GetStruct = driver.configure.signaling.nradio.cell.bwp.harq.downlink.
↪user.retransm.rb.get(cell_name = 'abc', index = 1, bwParts = repcap.BwParts.
↪Default)
```

Configures the number of RB and start RB for a certain retransmission, for user-defined DL HARQ, for BWP <bb>. Only relevant for disabled auto RIV.

param cell_name
No help available

param index
Index of the retransmission

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *index: int*, *nrb: int*, *start_rb: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:DL:USER:RETRansm:RB
driver.configure.signaling.nradio.cell.bwp.harq.downlink.user.retransm.rb.
↪set(cell_name = 'abc', index = 1, nrb = 1, start_rb = 1, bwParts = repcap.
↪BwParts.Default)
```

Configures the number of RB and start RB for a certain retransmission, for user-defined DL HARQ, for BWP <bb>. Only relevant for disabled auto RIV.

param cell_name
No help available

param index
Index of the retransmission

param nrb
No help available

param start_rb
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.97 Rversion

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:DL:USER:RETRansm:RVERsion
```

class RversionCls

Rversion commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *index: int*, *bwParts=BwParts.Default*) → Version

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:HARQ:DL:USER:RETRansm:RVERsion
value: enums.Version = driver.configure.signaling.nradio.cell.bwp.harq.downlink.
↪user.retransm.rversion.get(cell_name = 'abc', index = 1, bwParts = repcap.
↪BwParts.Default)
```

Selects a redundancy version for a certain retransmission, for user-defined DL HARQ, for BWP <bb>.

param cell_name
No help available

param index
Index of the retransmission

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
version: Auto mode, redundancy version number 0 to 3.

set(*cell_name: str*, *index: int*, *version: Version*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:HARQ:DL:USER:RETRansm:RVERsion
driver.configure.signaling.nradio.cell.bwp.harq.downlink.user.retransm.rversion.
↪set(cell_name = 'abc', index = 1, version = enums.Version.AUTO, bwParts =␣
↪repcap.BwParts.Default)
```

Selects a redundancy version for a certain retransmission, for user-defined DL HARQ, for BWP <bb>.

param cell_name
No help available

param index
Index of the retransmission

param version
Auto mode, redundancy version number 0 to 3.

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.98 Uplink

class UplinkCls

Uplink commands group definition. 9 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.harq.uplink.clone()
```

Subgroups

6.3.4.10.2.99 Auto

class AutoCls

Auto commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.harq.uplink.auto.clone()
```

Subgroups

6.3.4.10.2.100 Mret

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:UL:AUTO:MRET
```

class MretCls

Mret commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:UL:AUTO:MRET
value: int = driver.configure.signaling.nradio.cell.bwp.harq.uplink.auto.mret.
↪get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Configures the maximum number of retransmissions, for auto-configured UL HARQ, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
retransmissions: No help available

set(*cell_name: str*, *retransmissions: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:UL:AUTO:MRET
driver.configure.signaling.nradio.cell.bwp.harq.uplink.auto.mret.set(cell_name␣
↪= 'abc', retransmissions = 1, bwParts = repcap.BwParts.Default)
```

Configures the maximum number of retransmissions, for auto-configured UL HARQ, for BWP <bb>.

param cell_name
No help available

param retransmissions
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.101 Behavior

class BehaviorCls

Behavior commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.harq.uplink.behavior.clone()
```

Subgroups

6.3.4.10.2.102 CrcPass

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:UL:BEHavior:CRCPass
```

class CrcPassCls

CrcPass commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → AckOrDtx

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:UL:BEHavior:CRCPass
value: enums.AckOrDtx = driver.configure.signaling.nradio.cell.bwp.harq.uplink.
↪behavior.crcPass.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Defines the behavior when a UL transmission passes the CRC check: stop or continue requesting retransmissions from the UE, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
behavior: No help available

set(*cell_name: str*, *behavior: AckOrDtx*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:UL:BEHavior:CRCPass
driver.configure.signaling.nradio.cell.bwp.harq.uplink.behavior.crcPass.
↪set(cell_name = 'abc', behavior = enums.AckOrDtx.CONTinue, bwParts = repcap.
↪BwParts.Default)
```

Defines the behavior when a UL transmission passes the CRC check: stop or continue requesting retransmissions from the UE, for BWP <bb>.

param cell_name
No help available

param behavior
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.103 NulPower

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:UL:BEHavior:NULPower
```

class NulPowerCls

NulPower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → AckOrDtx

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:UL:BEHavior:NULPower
value: enums.AckOrDtx = driver.configure.signaling.nradio.cell.bwp.harq.uplink.
↪behavior.nulPower.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Defines the behavior when detecting no UL power: stop or continue requesting retransmissions from the UE, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
behavior: No help available

set(*cell_name: str*, *behavior: AckOrDtx*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:UL:BEHavior:NULPower
driver.configure.signaling.nradio.cell.bwp.harq.uplink.behavior.nulPower.
↪set(cell_name = 'abc', behavior = enums.AckOrDtx.CONTinue, bwParts = repcap.
↪BwParts.Default)
```

Defines the behavior when detecting no UL power: stop or continue requesting retransmissions from the UE, for BWP <bb>.

param cell_name
No help available

param behavior
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.104 Cmode

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:UL:CMODe
```

class CmodeCls

Cmode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → ModeC

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:UL:CMODe
value: enums.ModeC = driver.configure.signaling.nradio.cell.bwp.harq.uplink.
↪cmode.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Selects a mode for UL HARQ configuration, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
mode: NOTC: no UL HARQ AUTO: automatic configuration of the UL HARQ settings USER: user-defined configuration of the UL HARQ settings

set(*cell_name: str*, *mode: ModeC*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:UL:CMODe
driver.configure.signaling.nradio.cell.bwp.harq.uplink.cmode.set(cell_name =
↪'abc', mode = enums.ModeC.AUTO, bwParts = repcap.BwParts.Default)
```

Selects a mode for UL HARQ configuration, for BWP <bb>.

param cell_name
No help available

param mode
NOTC: no UL HARQ AUTO: automatic configuration of the UL HARQ settings USER: user-defined configuration of the UL HARQ settings

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.105 User

class UserCls

User commands group definition. 5 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.harq.uplink.user.clone()
```

Subgroups

6.3.4.10.2.106 Retransm

class RetransmCls

Retransm commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.harq.uplink.user.retransm.clone()
```

Subgroups

6.3.4.10.2.107 Ariv

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:UL:USER:RETRansm:ARIV
```

class ArivCls

Ariv commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *index: int*, *bwParts=BwParts.Default*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:UL:USER:RETRansm:ARIV
value: bool = driver.configure.signaling.nradio.cell.bwp.harq.uplink.user.
↪retransm.ariv.get(cell_name = 'abc', index = 1, bwParts = repcap.BwParts.
↪Default)
```

Configures auto RIV for a certain retransmission, for user-defined UL HARQ, for BWP <bb>.

param cell_name
No help available

param index
Index of the retransmission

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
riv: ON: Auto RIV enabled, no. of RB and start RB set automatically. OFF: Auto RIV disabled, you can define no. of RB and start RB.

set(*cell_name: str*, *index: int*, *riv: bool*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:UL:USER:RETRansm:ARIV
driver.configure.signaling.nradio.cell.bwp.harq.uplink.user.retransm.ariv.
↪set(cell_name = 'abc', index = 1, riv = False, bwParts = repcap.BwParts.
↪Default)
```

Configures auto RIV for a certain retransmission, for user-defined UL HARQ, for BWP <bb>.

param cell_name
No help available

param index
Index of the retransmission

param riv
ON: Auto RIV enabled, no. of RB and start RB set automatically. OFF: Auto RIV disabled, you can define no. of RB and start RB.

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.108 Modulation

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:UL:USER:RETRansm:MODulation
```

class ModulationCls

Modulation commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *index: int*, *bwParts=BwParts.Default*) → ModulationRetr

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:HARQ:UL:USER:RETRansm:MODulation
value: enums.ModulationRetr = driver.configure.signaling.nradio.cell.bwp.harq.
↪uplink.user.retransm.modulation.get(cell_name = 'abc', index = 1, bwParts =␣
↪repcap.BwParts.Default)
```

Selects a modulation scheme for a certain retransmission, for user-defined UL HARQ, for BWP <bb>.

param cell_name
No help available

param index
Index of the retransmission

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
modulation: BPSK, auto mode, /2-BPSK, QPSK, 16QAM, 64QAM, 256QAM, 1024QAM

set(*cell_name: str*, *index: int*, *modulation: ModulationRetr*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:HARQ:UL:USER:RETRansm:MODulation
driver.configure.signaling.nradio.cell.bwp.harq.uplink.user.retransm.modulation.
↪set(cell_name = 'abc', index = 1, modulation = enums.ModulationRetr.AUTO,␣
↪bwParts = repcap.BwParts.Default)
```

Selects a modulation scheme for a certain retransmission, for user-defined UL HARQ, for BWP <bb>.

param cell_name
No help available

param index
Index of the retransmission

param modulation
BPSK, auto mode, /2-BPSK, QPSK, 16QAM, 64QAM, 256QAM, 1024QAM

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.109 Moffset

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:UL:USER:RETRansm:MOFFset
```

class MoffsetCls

Moffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *index: int*, *bwParts=BwParts.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:HARQ:UL:USER:RETRansm:MOFFset
value: int = driver.configure.signaling.nradio.cell.bwp.harq.uplink.user.
↪retransm.moffset.get(cell_name = 'abc', index = 1, bwParts = repcap.BwParts.
↪Default)
```

Minimum number of slots between feedback processing and sending the retransmission DCI, for BWP <bb>.

param cell_name
No help available

param index
Index of the retransmission

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
minimum_offset: No help available

set(*cell_name: str*, *index: int*, *minimum_offset: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:HARQ:UL:USER:RETRansm:MOFFset
driver.configure.signaling.nradio.cell.bwp.harq.uplink.user.retransm.moffset.
↪set(cell_name = 'abc', index = 1, minimum_offset = 1, bwParts = repcap.
↪BwParts.Default)
```

Minimum number of slots between feedback processing and sending the retransmission DCI, for BWP <bb>.

param cell_name
No help available

param index
Index of the retransmission

param minimum_offset
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.110 Rb

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:UL:USER:RETRansm:RB
```

class RbCls

Rb commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Nrb: int: No parameter help available
- Start_Rb: int: No parameter help available

get(*cell_name: str*, *index: int*, *bwParts=BwParts.Default*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:UL:USER:RETRansm:RB
value: GetStruct = driver.configure.signaling.nradio.cell.bwp.harq.uplink.user.
↪retransm.rb.get(cell_name = 'abc', index = 1, bwParts = repcap.BwParts.
↪Default)
```

Configures the number of RB and start RB for a certain retransmission, for user-defined UL HARQ, for BWP <bb>. Only relevant for disabled auto RIV.

param cell_name
No help available

param index
Index of the retransmission

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *index: int*, *nrb: int*, *start_rb: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:UL:USER:RETRansm:RB
driver.configure.signaling.nradio.cell.bwp.harq.uplink.user.retransm.rb.
↪set(cell_name = 'abc', index = 1, nrb = 1, start_rb = 1, bwParts = repcap.
↪BwParts.Default)
```

Configures the number of RB and start RB for a certain retransmission, for user-defined UL HARQ, for BWP <bb>. Only relevant for disabled auto RIV.

param cell_name
No help available

param index
Index of the retransmission

param nrb
No help available

param start_rb
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.111 Rversion

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:UL:USER:RETRansm:RVERsion
```

class RversionCls

Rversion commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *index: int*, *bwParts=BwParts.Default*) → Version

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:HARQ:UL:USER:RETRansm:RVERsion
value: enums.Version = driver.configure.signaling.nradio.cell.bwp.harq.uplink.
↪user.retransm.rversion.get(cell_name = 'abc', index = 1, bwParts = repcap.
↪BwParts.Default)
```

Selects a redundancy version for a certain retransmission, for user-defined UL HARQ, for BWP <bb>.

param cell_name
No help available

param index
Index of the retransmission

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
version: Auto mode, redundancy version number 0 to 3.

set(*cell_name: str*, *index: int*, *version: Version*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:HARQ:UL:USER:RETRansm:RVERsion
driver.configure.signaling.nradio.cell.bwp.harq.uplink.user.retransm.rversion.
↪set(cell_name = 'abc', index = 1, version = enums.Version.AUTO, bwParts =␣
↪repcap.BwParts.Default)
```

Selects a redundancy version for a certain retransmission, for user-defined UL HARQ, for BWP <bb>.

param cell_name
No help available

param index
Index of the retransmission

param version
Auto mode, redundancy version number 0 to 3.

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.112 Nssb

class NssbCls

Nssb commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.nssb.clone()
```

Subgroups

6.3.4.10.2.113 Arfcn

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:NSSB:ARFCn
```

class ArfcnCls

Arfcn commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:NSSB:ARFCn
value: int = driver.configure.signaling.nradio.cell.bwp.nssb.arfcn.get(cell_
↪name = 'abc', bwParts = repcap.BwParts.Default)
```

Configures the channel number of the NCD-SSB signaled as 'absoluteFrequencySSB-r17', for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
number: No help available

set(*cell_name: str*, *number: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:NSSB:ARFCn
driver.configure.signaling.nradio.cell.bwp.nssb.arfcn.set(cell_name = 'abc',␣
↪number = 1, bwParts = repcap.BwParts.Default)
```

Configures the channel number of the NCD-SSB signaled as 'absoluteFrequencySSB-r17', for BWP <bb>.

param cell_name
No help available

param number
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.114 Enable

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:NSSB:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:NSSB:ENABle
value: bool = driver.configure.signaling.nradio.cell.bwp.nssb.enable.get(cell_
↪name = 'abc', bwParts = repcap.BwParts.Default)
```

Enables sending the IE 'NonCellDefiningSSB-r17', for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
enable: No help available

set(*cell_name: str*, *enable: bool*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:NSSB:ENABle
driver.configure.signaling.nradio.cell.bwp.nssb.enable.set(cell_name = 'abc',␣
↪enable = False, bwParts = repcap.BwParts.Default)
```

Enables sending the IE 'NonCellDefiningSSB-r17', for BWP <bb>.

param cell_name
No help available

param enable
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.115 Offset

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:NSSB:OFFSet
```

class OffsetCls

Offset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → TimeOffset

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:NSSB:OFFSet
value: enums.TimeOffset = driver.configure.signaling.nradio.cell.bwp.nssb.
↪offset.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Configures the time offset between the first burst of the NCD-SSB and the first burst of the CD-SSB, signaled as 'ssb-TimeOffset-r17', for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
time_offset: No help available

set(*cell_name: str*, *time_offset: TimeOffset*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:NSSB:OFFSet
driver.configure.signaling.nradio.cell.bwp.nssb.offset.set(cell_name = 'abc',␣
↪time_offset = enums.TimeOffset.T0, bwParts = repcap.BwParts.Default)
```

Configures the time offset between the first burst of the NCD-SSB and the first burst of the CD-SSB, signaled as 'ssb-TimeOffset-r17', for BWP <bb>.

param cell_name
No help available

param time_offset
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.116 Periodicity

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:NSSB:PERiodicity
```

class PeriodicityCls

Periodicity commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → PeriodicityB

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:NSSB:PERiodicity
value: enums.PeriodicityB = driver.configure.signaling.nradio.cell.bwp.nssb.
↪periodicity.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Configures the periodicity of the NCD-SSB signaled as 'ssb-Periodicity-r17', for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
periodicity: No help available

`set`(*cell_name: str*, *periodicity: PeriodicityB*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:NSSB:PERiodicity
driver.configure.signaling.nradio.cell.bwp.nssb.periodicity.set(cell_name = 'abc
↪', periodicity = enums.PeriodicityB.P10, bwParts = repcap.BwParts.Default)
```

Configures the periodicity of the NCD-SSB signaled as 'ssb-Periodicity-r17', for BWP <bb>.

param cell_name
No help available

param periodicity
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.117 Power

class PowerCls

Power commands group definition. 12 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.power.clone()
```

Subgroups

6.3.4.10.2.118 Control

class ControlCls

Control commands group definition. 12 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.power.control.clone()
```

Subgroups

6.3.4.10.2.119 Channel

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:POWer:CONTrol:CHANnel
```

class ChannelCls

Channel commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → SrcType

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:POWer:CONTrol:CHANnel
value: enums.SrcType = driver.configure.signaling.nradio.cell.bwp.power.control.
↪channel.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Selects the uplink channel types to which the power control commands are applied, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
type_py: PUSC: PUSCH PUCC: PUCCH PUPU: PUSCH and PUCCH

set(*cell_name: str*, *type_py: SrcType*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:POWer:CONTrol:CHANnel
driver.configure.signaling.nradio.cell.bwp.power.control.channel.set(cell_name␣
↪= 'abc', type_py = enums.SrcType.PUCC, bwParts = repcap.BwParts.Default)
```

Selects the uplink channel types to which the power control commands are applied, for BWP <bb>.

param cell_name
No help available

param type_py
PUSC: PUSCH PUCC: PUCCH PUPU: PUSCH and PUCCH

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.120 PalphaSet

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:POWer:CONTrol:PALPhaset
```

class PalphaSetCls

PalphaSet commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Enable: bool: ON: Signal the 'P0-PUSCH-AlphaSet'. OFF: Do not signal the 'P0-PUSCH-AlphaSet'.
- Alpha: enums.Alpha: No parameter help available
- P_0: int: No parameter help available

get(*cell_name: str*, *bwParts=BwParts.Default*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:POWer:CONTrol:PALPhaset
value: GetStruct = driver.configure.signaling.nradio.cell.bwp.power.control.
↪palphaSet.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Sets the parameters 'alpha' and 'p0' of the 'P0-PUSCH-AlphaSet' that is signaled to the UE, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *enable: bool*, *alpha: Alpha = None*, *p_0: int = None*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:POWer:CONTrol:PALPhaset
driver.configure.signaling.nradio.cell.bwp.power.control.palphaSet.set(cell_
↪name = 'abc', enable = False, alpha = enums.Alpha.A00, p_0 = 1, bwParts =␣
↪repcap.BwParts.Default)
```

Sets the parameters 'alpha' and 'p0' of the 'P0-PUSCH-AlphaSet' that is signaled to the UE, for BWP <bb>.

param cell_name
No help available

param enable
ON: Signal the 'P0-PUSCH-AlphaSet'. OFF: Do not signal the 'P0-PUSCH-AlphaSet'.

param alpha
No help available

param p_0
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.121 PbpiBpsk

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:POWer:CONTrol:PBPibpsk
```

class PbpiBpskCls

PbpiBpsk commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:POWer:CONTrol:PBPibpsk
value: bool = driver.configure.signaling.nradio.cell.bwp.power.control.pbpiBpsk.
↪get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Configures the signaled optional Boolean value 'powerBoostPi2BPSK', for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
enable: No help available

set(*cell_name: str*, *enable: bool*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:POWer:CONTrol:PBPibpsk
driver.configure.signaling.nradio.cell.bwp.power.control.pbpiBpsk.set(cell_name␣
↪= 'abc', enable = False, bwParts = repcap.BwParts.Default)
```

Configures the signaled optional Boolean value 'powerBoostPi2BPSK', for BWP <bb>.

param cell_name
No help available

param enable
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.122 TpControl

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:POWer:CONTrol:TPControl
```

class TpControlCls

TpControl commands group definition. 9 total commands, 3 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → TpControl

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:POWer:CONTrol:TPControl
value: enums.TpControl = driver.configure.signaling.nradio.cell.bwp.power.
↪control.tpControl.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Selects the pattern of TPC commands to be sent to the UE, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
control: Keep, min, max, closed loop, TPC pattern, relative power tolerance.

set(*cell_name: str*, *control: TpControl*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:POWer:CONTrol:TPControl
driver.configure.signaling.nradio.cell.bwp.power.control.tpControl.set(cell_
↪name = 'abc', control = enums.TpControl.CLOop, bwParts = repcap.BwParts.
↪Default)
```

Selects the pattern of TPC commands to be sent to the UE, for BWP <bb>.

param cell_name
No help available

param control
Keep, min, max, closed loop, TPC pattern, relative power tolerance.

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.power.control.tpControl.clone()
```

Subgroups

6.3.4.10.2.123 Cloop

class CloopCls

Cloop commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.power.control.tpControl.cloop.clone()
```

Subgroups

6.3.4.10.2.124 Tolerance

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:POWer:CONTrol:TPControl:CLOop:TOLerance
```

class ToleranceCls

Tolerance commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → float

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>
↪:POWer:CONTrol:TPControl:CLOop:TOLerance
value: float = driver.configure.signaling.nradio.cell.bwp.power.control.
↪tpControl.cloop.tolerance.get(cell_name = 'abc', bwParts = repcap.BwParts.
↪Default)
```

Defines the tolerance for closed-loop power control for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
tolerance: No help available

set(*cell_name: str*, *tolerance: float*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>
↪:POWer:CONTrol:TPControl:CLOop:TOLerance
driver.configure.signaling.nradio.cell.bwp.power.control.tpControl.cloop.
↪tolerance.set(cell_name = 'abc', tolerance = 1.0, bwParts = repcap.BwParts.
↪Default)
```

Defines the tolerance for closed-loop power control for BWP <bb>.

param cell_name
No help available

param tolerance
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.125 Tpower

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:POWer:CONTrol:TPControl:CLOop:TPOWer
```

class TpowerCls

Tpower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → float

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>
↪:POWer:CONTrol:TPControl:CLOop:TPOWer
value: float = driver.configure.signaling.nradio.cell.bwp.power.control.
↪tpControl.cloop.tpower.get(cell_name = 'abc', bwParts = repcap.BwParts.
↪Default)
```

Defines the target power for closed-loop power control, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
power: No help available

set(*cell_name: str*, *power: float*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>
↪:POWer:CONTrol:TPControl:CLOop:TPOWer
driver.configure.signaling.nradio.cell.bwp.power.control.tpControl.cloop.tpower.
↪set(cell_name = 'abc', power = 1.0, bwParts = repcap.BwParts.Default)
```

Defines the target power for closed-loop power control, for BWP <bb>.

param cell_name
No help available

param power
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.126 Pattern

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:POWer:CONTrol:TPControl:PATTern
```

class PatternCls

Pattern commands group definition. 3 total commands, 1 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → TypeB

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>
↪:POWer:CONTrol:TPControl:PATTern
value: enums.TypeB = driver.configure.signaling.nradio.cell.bwp.power.control.
↪tpControl.pattern.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

No command help available

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
type_py: No help available

set(*cell_name: str*, *type_py: TypeB*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>
↪:POWer:CONTrol:TPControl:PATTern
driver.configure.signaling.nradio.cell.bwp.power.control.tpControl.pattern.
↪set(cell_name = 'abc', type_py = enums.TypeB.UDEFined, bwParts = repcap.
↪BwParts.Default)
```

No command help available

param cell_name
No help available

param type_py
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.power.control.tpControl.pattern.
↪clone()
```

Subgroups

6.3.4.10.2.127 UserDefined

class UserDefinedCls

UserDefined commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.power.control.tpControl.pattern.
↪userDefined.clone()
```

Subgroups

6.3.4.10.2.128 Mode

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>
↪:POWer:CONTrol:TPControl:PATTern:UDEFined:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → Repeat

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>
↪:POWer:CONTrol:TPControl:PATTern:UDEFined:MODE
value: enums.Repeat = driver.configure.signaling.nradio.cell.bwp.power.control.
↪tpControl.pattern.userDefined.mode.get(cell_name = 'abc', bwParts = repcap.
↪BwParts.Default)
```

Selects the mode for execution of a user-defined TPC pattern, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
mode: No help available

set(*cell_name: str*, *mode: Repeat*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>
↪:POWer:CONTrol:TPControl:PATTern:UDEFined:MODE
driver.configure.signaling.nradio.cell.bwp.power.control.tpControl.pattern.
↪userDefined.mode.set(cell_name = 'abc', mode = enums.Repeat.CONTinuous,␣
↪bwParts = repcap.BwParts.Default)
```

Selects the mode for execution of a user-defined TPC pattern, for BWP <bb>.

param cell_name
No help available

param mode
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.129 Pattern

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>
↪:POWer:CONTrol:TPControl:PATTern:UDEFined:PATTern
```

class PatternCls

Pattern commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → List[Pattern]

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>
↪:POWer:CONTrol:TPControl:PATTern:UDEFined:PATTern
value: List[enums.Pattern] = driver.configure.signaling.nradio.cell.bwp.power.
↪control.tpControl.pattern.userDefined.pattern.get(cell_name = 'abc', bwParts␣
↪= repcap.BwParts.Default)
```

Configures a user-defined TPC pattern as a sequence of commands, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
pattern: Comma-separated list of commands D1: -1 dB KEEP: 0 dB U1: +1 dB U3: +3 dB A single NAV is returned if no pattern is defined.

set(*cell_name: str*, *pattern: List[Pattern]*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>
↪:POWer:CONTrol:TPControl:PATTern:UDEFined:PATTern
```

(continues on next page)

(continued from previous page)

```
driver.configure.signaling.nradio.cell.bwp.power.control.tpControl.pattern.
↪userDefined.pattern.set(cell_name = 'abc', pattern = [Pattern.D1, Pattern.U3],
↪ bwParts = repcap.BwParts.Default)
```

Configures a user-defined TPC pattern as a sequence of commands, for BWP <bb>.

param cell_name
No help available

param pattern
Comma-separated list of commands D1: -1 dB KEEP: 0 dB U1: +1 dB U3: +3 dB A single NAV is returned if no pattern is defined.

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.130 RpTolerance

class RpToleranceCls

RpTolerance commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.power.control.tpControl.rpTolerance.
↪clone()
```

Subgroups

6.3.4.10.2.131 Direction

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>
↪:POWer:CONTrol:TPControl:RPTolerance:DIRection
```

class DirectionCls

Direction commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → TpcDirection

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>
↪:POWer:CONTrol:TPControl:RPTolerance:DIRection
value: enums.TpcDirection = driver.configure.signaling.nradio.cell.bwp.power.
↪control.tpControl.rpTolerance.direction.get(cell_name = 'abc', bwParts =␣
↪repcap.BwParts.Default)
```

Selects the direction of the TPC pattern for relative power tolerance tests, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
direction: No help available

set(*cell_name: str*, *direction: TpcDirection*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>
↪:POWer:CONTrol:TPControl:RPTolerance:DIRection
driver.configure.signaling.nradio.cell.bwp.power.control.tpControl.rpTolerance.
↪direction.set(cell_name = 'abc', direction = enums.TpcDirection.ALTernating,␣
↪bwParts = repcap.BwParts.Default)
```

Selects the direction of the TPC pattern for relative power tolerance tests, for BWP <bb>.

param cell_name
No help available

param direction
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.132 Pattern

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:POWer:CONTrol:TPControl:RPTolerance:PATTern
```

class PatternCls

Pattern commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → RpPattern

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>
↪:POWer:CONTrol:TPControl:RPTolerance:PATTern
value: enums.RpPattern = driver.configure.signaling.nradio.cell.bwp.power.
↪control.tpControl.rpTolerance.pattern.get(cell_name = 'abc', bwParts = repcap.
↪BwParts.Default)
```

Selects a TPC pattern for ramping up and ramping down relative power tolerance tests, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
pattern: No help available

set(*cell_name: str*, *pattern: RpPattern*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>
↪:POWer:CONTrol:TPControl:RPTolerance:PATTern
driver.configure.signaling.nradio.cell.bwp.power.control.tpControl.rpTolerance.
↪pattern.set(cell_name = 'abc', pattern = enums.RpPattern.A, bwParts = repcap.
↪BwParts.Default)
```

Selects a TPC pattern for ramping up and ramping down relative power tolerance tests, for BWP <bb>.

param cell_name
No help available

param pattern
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.133 StId

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:POWer:CONTrol:TPControl:RPTolerance:STID
```

class StIdCls

StId commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>
↪:POWer:CONTrol:TPControl:RPTolerance:STID
value: int = driver.configure.signaling.nradio.cell.bwp.power.control.tpControl.
↪rpTolerance.stId.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Selects the subtest ID for relative power tolerance tests, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
sub_test_id: No help available

set(*cell_name: str*, *sub_test_id: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>
↪:POWer:CONTrol:TPControl:RPTolerance:STID
driver.configure.signaling.nradio.cell.bwp.power.control.tpControl.rpTolerance.
↪stId.set(cell_name = 'abc', sub_test_id = 1, bwParts = repcap.BwParts.Default)
```

Selects the subtest ID for relative power tolerance tests, for BWP <bb>.

param cell_name
No help available

param sub_test_id
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.134 Pucch

class PucchCls

Pucch commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.pucch.clone()
```

Subgroups

6.3.4.10.2.135 FormatPy

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:PUCCh:FORMat
```

class FormatPyCls

FormatPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → CellPucchFormatPy

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:PUCCh:FORMat
value: enums.CellPucchFormatPy = driver.configure.signaling.nradio.cell.bwp.
↪pucch.formatPy.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Selects the PUCCH format for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
format_py: No help available

set(*cell_name: str*, *format_py: CellPucchFormatPy*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:PUCCh:FORMat
driver.configure.signaling.nradio.cell.bwp.pucch.formatPy.set(cell_name = 'abc',
↪ format_py = enums.CellPucchFormatPy.F0, bwParts = repcap.BwParts.Default)
```

Selects the PUCCH format for BWP <bb>.

param cell_name
No help available

param format_py
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.136 Mode

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:PUCCh:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → ConfigMode

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:PUCCh:MODE
value: enums.ConfigMode = driver.configure.signaling.nradio.cell.bwp.pucch.mode.
↪get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Selects the configuration mode for the PUCCH resources, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
mode: No help available

set(*cell_name: str*, *mode: ConfigMode*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:PUCCh:MODE
driver.configure.signaling.nradio.cell.bwp.pucch.mode.set(cell_name = 'abc',␣
↪mode = enums.ConfigMode.AUTO, bwParts = repcap.BwParts.Default)
```

Selects the configuration mode for the PUCCH resources, for BWP <bb>.

param cell_name
No help available

param mode
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.137 Papr

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:PUCCh:PAPR
```

class PaprCls

Papr commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:PUCCh:PAPR
value: bool = driver.configure.signaling.nradio.cell.bwp.pucch.papr.get(cell_
↪name = 'abc', bwParts = repcap.BwParts.Default)
```

Enables the usage of a DMRS with a low PAPR, for PUCCH, BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
enable: No help available

set(*cell_name: str*, *enable: bool*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:PUCCh:PAPR
driver.configure.signaling.nradio.cell.bwp.pucch.papr.set(cell_name = 'abc',␣
↪enable = False, bwParts = repcap.BwParts.Default)
```

Enables the usage of a DMRS with a low PAPR, for PUCCH, BWP <bb>.

param cell_name
No help available

param enable
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.138 Sprb

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:PUCCh:SPRB
```

class SprbCls

Sprb commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → LowHigh

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:PUCCh:SPRB
value: enums.LowHigh = driver.configure.signaling.nradio.cell.bwp.pucch.sprb.
↪get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Selects the position of the resource blocks: lower end or upper end of the allowed range. For BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
starting_prb: No help available

set(*cell_name: str*, *starting_prb: LowHigh*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:PUCCh:SPRB
driver.configure.signaling.nradio.cell.bwp.pucch.sprb.set(cell_name = 'abc',␣
↪starting_prb = enums.LowHigh.HIGH, bwParts = repcap.BwParts.Default)
```

Selects the position of the resource blocks: lower end or upper end of the allowed range. For BWP <bb>.

param cell_name
No help available

param starting_prb
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.139 Pusch

class PuschCls

Pusch commands group definition. 6 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.pusch.clone()
```

Subgroups

6.3.4.10.2.140 Dtfs

class DtfsCls

Dtfs commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.pusch.dtfs.clone()
```

Subgroups

6.3.4.10.2.141 McsTable

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:PUSCh:DTFS:MCSTable
```

class McsTableCls

McsTable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → McsTableB

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:PUSCh:DTFS:MCSTable
value: enums.McsTableB = driver.configure.signaling.nradio.cell.bwp.pusch.dtfs.
↪mcsTable.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Defines which MCS table must be used for PUSCH with transform precoding (parameter 'mcsTableTransformPrecoder') , for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
mcs: 256QAM, 64QAM low SE, 64QAM

set(*cell_name: str*, *mcs: McsTableB*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:PUSCh:DTFS:MCSTable
driver.configure.signaling.nradio.cell.bwp.pusch.dtfs.mcsTable.set(cell_name =
↪'abc', mcs = enums.McsTableB.L64, bwParts = repcap.BwParts.Default)
```

Defines which MCS table must be used for PUSCH with transform precoding (parameter 'mcsTableTransformPrecoder') , for BWP <bb>.

param cell_name
No help available

param mcs
256QAM, 64QAM low SE, 64QAM

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.142 PibPsk

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:PUSCh:DTFS:PIBPsk
```

class PibPskCls

PibPsk commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:PUSCh:DTFS:PIBPsk
value: bool = driver.configure.signaling.nradio.cell.bwp.pusch.dtfs.pibPsk.
↪get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Enables the modulation scheme /2-BPSK for PUSCH with transform precoding, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
enable: No help available

set(*cell_name: str*, *enable: bool*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:PUSCh:DTFS:PIBPsk
driver.configure.signaling.nradio.cell.bwp.pusch.dtfs.pibPsk.set(cell_name =
↪'abc', enable = False, bwParts = repcap.BwParts.Default)
```

Enables the modulation scheme /2-BPSK for PUSCH with transform precoding, for BWP <bb>.

param cell_name
No help available

param enable
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.143 TpRecoding

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:PUSCh:TPRecoding
```

class TpRecodingCls

TpRecoding commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → Waveform

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:PUSCh:TPRecoding
value: enums.Waveform = driver.configure.signaling.nradio.cell.bwp.pusch.
↪tpRecoding.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Defines which type of OFDM the UE must use for the PUSCH, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
waveform: CP: CP-OFDM (no transform precoding) . DTFS: DFT-s-OFDM (with transform precoding) .

set(*cell_name: str*, *waveform: Waveform*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:PUSCh:TPRecoding
driver.configure.signaling.nradio.cell.bwp.pusch.tpRecoding.set(cell_name = 'abc
↪', waveform = enums.Waveform.CP, bwParts = repcap.BwParts.Default)
```

Defines which type of OFDM the UE must use for the PUSCH, for BWP <bb>.

param cell_name
No help available

param waveform
CP: CP-OFDM (no transform precoding) . DTFS: DFT-s-OFDM (with transform precoding) .

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.144 Tschema

class TschemaCls

Tschema commands group definition. 3 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.pusch.tschema.clone()
```

Subgroups

6.3.4.10.2.145 Codebook

class CodebookCls

Codebook commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.pusch.tschema.codebook.clone()
```

Subgroups

6.3.4.10.2.146 FptMode

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:PUSCh:TSCHema:CODebook:FPTMode
```

class FptModeCls

FptMode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → FtpMode

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>
↪:PUSCh:TSCHema:CODebook:FPTMode
value: enums.FtpMode = driver.configure.signaling.nradio.cell.bwp.pusch.tschema.
↪codebook.fptMode.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Selects the signaled 'ul-FullPowerTransmission-r16', for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
mode: AUTO: signaled value selected via reported UE capabilities FULL: signaled value 'fullpower' MOD1: signaled value 'fullpowerMode1' MOD2: signaled value 'fullpowerMode2' OFF: 'ul-FullPowerTransmission-r16' not signaled

set(*cell_name: str*, *mode: FtpMode*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>
↪:PUSCh:TSCHema:CODebook:FPTMode
driver.configure.signaling.nradio.cell.bwp.pusch.tschema.codebook.fptMode.
↪set(cell_name = 'abc', mode = enums.FtpMode.AUTO, bwParts = repcap.BwParts.
↪Default)
```

Selects the signaled 'ul-FullPowerTransmission-r16', for BWP <bb>.

param cell_name
No help available

param mode
AUTO: signaled value selected via reported UE capabilities FULL: signaled value 'fullpower' MOD1: signaled value 'fullpowerMode1' MOD2: signaled value 'fullpowerMode2' OFF: 'ul-FullPowerTransmission-r16' not signaled

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.147 Subset

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:PUSCh:TSCHema:CODebook:SUBSet
```

class SubsetCls

Subset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → CodebookSubset

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>
↪:PUSCh:TSCHema:CODebook:SUBSet
value: enums.CodebookSubset = driver.configure.signaling.nradio.cell.bwp.pusch.
↪tschema.codebook.subset.get(cell_name = 'abc', bwParts = repcap.BwParts.
↪Default)
```

Selects the codebook subset for codebook-based transmission (signaled 'codebookSubset') , for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
subset: AUTO: signaled value selected via reported UE capabilities FPNC: signaled value 'fullyAndPartialAndNonCoherent' PNC: signaled value 'partialAndNonCoherent', currently not supported NC: signaled value 'nonCoherent'

set(*cell_name: str*, *subset: CodebookSubset*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>
↪:PUSCh:TSCHema:CODebook:SUBSet
driver.configure.signaling.nradio.cell.bwp.pusch.tschema.codebook.subset.
↪set(cell_name = 'abc', subset = enums.CodebookSubset.AUTO, bwParts = repcap.
↪BwParts.Default)
```

Selects the codebook subset for codebook-based transmission (signaled 'codebookSubset') , for BWP <bb>.

param cell_name
No help available

param subset
AUTO: signaled value selected via reported UE capabilities FPNC: signaled value 'fullyAndPartialAndNonCoherent' PNC: signaled value 'partialAndNonCoherent', currently not supported NC: signaled value 'nonCoherent'

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.148 Mode

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:PUSCh:TSCHema:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → Choice

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:PUSCh:TSCHema:MODE
value: enums.Choice = driver.configure.signaling.nradio.cell.bwp.pusch.tschema.
↪mode.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Selects the PUSCH transmission scheme, signaled as 'txConfig', for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
choice: SINGle: single antenna port, 'txConfig' not signaled CODebook: codebook-based transmission NCODebook: currently not supported

set(*cell_name: str*, *choice: Choice*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bpwid>:PUSCh:TSCHema:MODE
driver.configure.signaling.nradio.cell.bwp.pusch.tschema.mode.set(cell_name =
↪'abc', choice = enums.Choice.CODebook, bwParts = repcap.BwParts.Default)
```

Selects the PUSCH transmission scheme, signaled as 'txConfig', for BWP <bb>.

param cell_name
No help available

param choice
SINGle: single antenna port, 'txConfig' not signaled CODebook: codebook-based transmission NCODebook: currently not supported

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.149 Smode

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP:SMODe
```

class SmodeCls

Smode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → BwpSwitchingMode

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP:SMODe
value: enums.BwpSwitchingMode = driver.configure.signaling.nradio.cell.bwp.
↪smode.get(cell_name = 'abc')
```

Selects a mechanism for sending BWP switching information to the UE.

param cell_name
No help available

return
switching_mode: - STATic: The BWP is switched via an RRC connection reconfiguration. - DYNamic: The BWP is switched by sending a BWP ID in DCI format 0_1 for the UL or format 1_1 for the DL.

set(*cell_name: str*, *switching_mode: BwpSwitchingMode*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP:SMODe
driver.configure.signaling.nradio.cell.bwp.smode.set(cell_name = 'abc',␣
↪switching_mode = enums.BwpSwitchingMode.DYNamic)
```

Selects a mechanism for sending BWP switching information to the UE.

param cell_name
No help available

param switching_mode

- STATic: The BWP is switched via an RRC connection reconfiguration.
- DYNamic: The BWP is switched by sending a BWP ID in DCI format 0_1 for the UL or format 1_1 for the DL.

6.3.4.10.2.150 Srs

class SrsCls

Srs commands group definition. 19 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.srs.clone()
```

Subgroups

6.3.4.10.2.151 Aswitching

class AswitchingCls

Aswitching commands group definition. 9 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.srs.aswitching.clone()
```

Subgroups

6.3.4.10.2.152 Enable

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:SRS:ASWitching:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:SRS:ASWitching:ENABle
value: bool = driver.configure.signaling.nradio.cell.bwp.srs.aswitching.enable.
↪get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Enables or disables SRS antenna switching for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
enable: No help available

set(*cell_name: str*, *enable: bool*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:SRS:ASWitching:ENABle
driver.configure.signaling.nradio.cell.bwp.srs.aswitching.enable.set(cell_name␣
↪= 'abc', enable = False, bwParts = repcap.BwParts.Default)
```

Enables or disables SRS antenna switching for BWP <bb>.

param cell_name
No help available

param enable
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.153 Power

class PowerCls

Power commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.srs.aswitching.power.clone()
```

Subgroups

6.3.4.10.2.154 Alpha

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:SRS:ASWitching:POWer:ALPHa
```

class AlphaCls

Alpha commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → Alpha

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:SRS:ASWitching:POWer:ALPHa
value: enums.Alpha = driver.configure.signaling.nradio.cell.bwp.srs.aswitching.
↪power.alpha.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Sets the SRS power control parameter 'alpha' for SRS antenna switching, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
alpha: Axy means x.y.

set(*cell_name: str*, *alpha: Alpha*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:SRS:ASWitching:POWer:ALPHa
driver.configure.signaling.nradio.cell.bwp.srs.aswitching.power.alpha.set(cell_
↪name = 'abc', alpha = enums.Alpha.A00, bwParts = repcap.BwParts.Default)
```

Sets the SRS power control parameter 'alpha' for SRS antenna switching, for BWP <bb>.

param cell_name
No help available

param alpha
Axy means x.y.

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.155 Pzero

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:SRS:ASWitching:POWer:PZERo
```

class PzeroCls

Pzero commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:SRS:ASWitching:POWer:PZERo
value: int = driver.configure.signaling.nradio.cell.bwp.srs.aswitching.power.
↪pzero.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Sets the SRS power control parameter 'p0' for SRS antenna switching, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
p_0: No help available

set(*cell_name: str*, *p_0: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:SRS:ASWitching:POWer:PZERo
driver.configure.signaling.nradio.cell.bwp.srs.aswitching.power.pzero.set(cell_
↪name = 'abc', p_0 = 1, bwParts = repcap.BwParts.Default)
```

Sets the SRS power control parameter 'p0' for SRS antenna switching, for BWP <bb>.

param cell_name
No help available

param p_0
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.156 Resource

class ResourceCls

Resource commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.srs.aswitching.resource.clone()
```

Subgroups

6.3.4.10.2.157 Fhopping

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:SRS:ASWitching:RESource:FHOPping
```

class FhoppingCls

Fhopping commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Csrs: int: No parameter help available
- Bsrs: int: No parameter help available
- Bhop: int: No parameter help available

get(*cell_name: str*, *resource_no: int*, *bwParts=BwParts.Default*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:SRS:ASWitching:RESource:FHOPping
value: GetStruct = driver.configure.signaling.nradio.cell.bwp.srs.aswitching.
↪resource.fhopping.get(cell_name = 'abc', resource_no = 1, bwParts = repcap.
↪BwParts.Default)
```

Configures the frequency hopping for SRS resource <ResourceNo> for SRS antenna switching, for BWP <bb>.

param cell_name
No help available

param resource_no
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *resource_no: int*, *csrs: int*, *bsrs: int = None*, *bhop: int = None*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:SRS:ASWitching:RESource:FHOPping
driver.configure.signaling.nradio.cell.bwp.srs.aswitching.resource.fhopping.
↪set(cell_name = 'abc', resource_no = 1, csrs = 1, bsrs = 1, bhop = 1, bwParts␣
↪= repcap.BwParts.Default)
```

Configures the frequency hopping for SRS resource <ResourceNo> for SRS antenna switching, for BWP <bb>.

param cell_name
No help available

param resource_no
No help available

param csrs
No help available

param bsrs
No help available

param bhop
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.158 Resource

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:SRS:ASWitching:RESource:RESource
```

class ResourceCls

Resource commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Resource_Id: int: No parameter help available
- No_Srs_Ports: enums.AntNoPorts: Number of antenna ports for SRS transmissions
- Fd_Position: int: Frequency domain position

- Fd_Shift: int: Frequency domain shift
- Sequence_Id: int: No parameter help available

class SetStruct

Structure for setting input parameters. Contains optional setting parameters. Fields:

- Cell_Name: str: No parameter help available
- Resource_No: int: No parameter help available
- Resource_Id: int: No parameter help available
- No_Srs_Ports: enums.AntNoPorts: Optional setting parameter. Number of antenna ports for SRS transmissions
- Fd_Position: int: Optional setting parameter. Frequency domain position
- Fd_Shift: int: Optional setting parameter. Frequency domain shift
- Sequence_Id: int: No parameter help available

get(*cell_name: str*, *resource_no: int*, *bwParts=BwParts.Default*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:SRS:ASWitching:RESource:RESource
value: GetStruct = driver.configure.signaling.nradio.cell.bwp.srs.aswitching.
↪resource.resource.get(cell_name = 'abc', resource_no = 1, bwParts = repcap.
↪BwParts.Default)
```

Configures the SRS resource <ResourceNo> for SRS antenna switching, for BWP <bb>.

param cell_name
No help available

param resource_no
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*structure: SetStruct*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:SRS:ASWitching:RESource:RESource
structure = driver.configure.signaling.nradio.cell.bwp.srs.aswitching.resource.
↪resource.SetStruct()
structure.Cell_Name: str = 'abc'
structure.Resource_No: int = 1
structure.Resource_Id: int = 1
structure.No_Srs_Ports: enums.AntNoPorts = enums.AntNoPorts.P1
structure.Fd_Position: int = 1
structure.Fd_Shift: int = 1
structure.Sequence_Id: int = 1
driver.configure.signaling.nradio.cell.bwp.srs.aswitching.resource.resource.
↪set(structure, bwParts = repcap.BwParts.Default)
```

Configures the SRS resource <ResourceNo> for SRS antenna switching, for BWP <bb>.

param structure
for set value, see the help for SetStruct structure arguments.

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.159 Rmapping

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:SRS:ASWitching:RESource:RMAPping
```

class RmappingCls

Rmapping commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Start_Position: int: No parameter help available
- No_Symbols: enums.NoSymbolsN: No parameter help available
- Rep_Factor: enums.NoSymbolsN: Repetition factor

get(*cell_name: str*, *resource_no: int*, *bwParts=BwParts.Default*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:SRS:ASWitching:RESource:RMAPping
value: GetStruct = driver.configure.signaling.nradio.cell.bwp.srs.aswitching.
↪resource.rmapping.get(cell_name = 'abc', resource_no = 1, bwParts = repcap.
↪BwParts.Default)
```

Configures the resource mapping for SRS transmissions, for SRS resource <ResourceNo> for SRS antenna switching, for BWP <bb>.

param cell_name
No help available

param resource_no
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *resource_no: int*, *start_position: int*, *no_symbols: NoSymbolsN = None*, *rep_factor: NoSymbolsN = None*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:SRS:ASWitching:RESource:RMAPping
driver.configure.signaling.nradio.cell.bwp.srs.aswitching.resource.rmapping.
```

(continues on next page)

(continued from previous page)

```
↪set(cell_name = 'abc', resource_no = 1, start_position = 1, no_symbols =␣
↪enums.NoSymbolsN.N1, rep_factor = enums.NoSymbolsN.N1, bwParts = repcap.
↪BwParts.Default)
```

Configures the resource mapping for SRS transmissions, for SRS resource <ResourceNo> for SRS antenna switching, for BWP <bb>.

param cell_name
No help available

param resource_no
No help available

param start_position
No help available

param no_symbols
No help available

param rep_factor
Repetition factor

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.160 Rtype

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:SRS:ASWitching:RESource:RTYPe
```

class RtypeCls

Rtype commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Period: int: Periodicity of slots (SRS every nth slot)
- Offset: int: Offset as number of slots. Must be smaller than the Period.

get(*cell_name: str*, *resource_no: int*, *bwParts=BwParts.Default*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:SRS:ASWitching:RESource:RTYPe
value: GetStruct = driver.configure.signaling.nradio.cell.bwp.srs.aswitching.
↪resource.rtype.get(cell_name = 'abc', resource_no = 1, bwParts = repcap.
↪BwParts.Default)
```

Configures the resource type for SRS resource <ResourceNo> for SRS antenna switching, for BWP <bb>.

param cell_name
No help available

param resource_no
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *resource_no: int*, *period: int = None*, *offset: int = None*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:SRS:ASWitching:RESource:RTYPe
driver.configure.signaling.nradio.cell.bwp.srs.aswitching.resource.rtype.
↪set(cell_name = 'abc', resource_no = 1, period = 1, offset = 1, bwParts =␣
↪repcap.BwParts.Default)
```

Configures the resource type for SRS resource <ResourceNo> for SRS antenna switching, for BWP <bb>.

param cell_name
No help available

param resource_no
No help available

param period
Periodicity of slots (SRS every nth slot)

param offset
Offset as number of slots. Must be smaller than the Period.

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.161 Tcomb

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:SRS:ASWitching:RESource:TCOMb
```

class TcombCls

Tcomb commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Ktc: enums.Ktc: No parameter help available
- Offset: int: No parameter help available
- Cyclic_Shift: int: No parameter help available

get(*cell_name: str*, *resource_no: int*, *bwParts=BwParts.Default*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:SRS:ASWitching:RESource:TCOMb
value: GetStruct = driver.configure.signaling.nradio.cell.bwp.srs.aswitching.
```

(continues on next page)

(continued from previous page)

```
↪resource.tcomb.get(cell_name = 'abc', resource_no = 1, bwParts = repcap.
↪BwParts.Default)
```

Configures the comb structure of the SRS resource <ResourceNo> for SRS antenna switching, for BWP <bb>.

param cell_name
No help available

param resource_no
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *resource_no: int*, *ktc: Ktc = None*, *offset: int = None*, *cyclic_shift: int = None*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:SRS:ASWitching:RESource:TCOMb
driver.configure.signaling.nradio.cell.bwp.srs.aswitching.resource.tcomb.
↪set(cell_name = 'abc', resource_no = 1, ktc = enums.Ktc.N2, offset = 1,␣
↪cyclic_shift = 1, bwParts = repcap.BwParts.Default)
```

Configures the comb structure of the SRS resource <ResourceNo> for SRS antenna switching, for BWP <bb>.

param cell_name
No help available

param resource_no
No help available

param ktc
No help available

param offset
No help available

param cyclic_shift
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.162 TypePy

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:SRS:ASWitching:TYPE
```

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → AswitchingType

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:SRS:ASWitching:TYPE
value: enums.AswitchingType = driver.configure.signaling.nradio.cell.bwp.srs.
↪aswitching.typePy.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Selects the antenna switching resource type, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
type_py: TtRr defines the number of ports t per SRS resource and the total number of ports over all SRS resources r.

set(*cell_name: str*, *type_py: AswitchingType*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:SRS:ASWitching:TYPE
driver.configure.signaling.nradio.cell.bwp.srs.aswitching.typePy.set(cell_name␣
↪= 'abc', type_py = enums.AswitchingType.T1R1, bwParts = repcap.BwParts.
↪Default)
```

Selects the antenna switching resource type, for BWP <bb>.

param cell_name
No help available

param type_py
TtRr defines the number of ports t per SRS resource and the total number of ports over all SRS resources r.

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.163 CnCodebook

class CnCodebookCls

CnCodebook commands group definition. 10 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.srs.cnCodebook.clone()
```

Subgroups

6.3.4.10.2.164 Power

class PowerCls

Power commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.srs.cnCodebook.power.clone()
```

Subgroups

6.3.4.10.2.165 Alpha

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:SRS:CNCodebook:POWer:ALPHa
```

class AlphaCls

Alpha commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → Alpha

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:SRS:CNCodebook:POWer:ALPHa
value: enums.Alpha = driver.configure.signaling.nradio.cell.bwp.srs.cnCodebook.
↪power.alpha.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Sets the SRS power control parameter ‘alpha’ for periodic SRS, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Bwp’)

return
alpha: Axy means x.y.

set(*cell_name: str*, *alpha: Alpha*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:SRS:CNCodebook:POWer:ALPHa
driver.configure.signaling.nradio.cell.bwp.srs.cnCodebook.power.alpha.set(cell_
↪name = 'abc', alpha = enums.Alpha.A00, bwParts = repcap.BwParts.Default)
```

Sets the SRS power control parameter 'alpha' for periodic SRS, for BWP <bb>.

param cell_name
No help available

param alpha
Axy means x.y.

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.166 Pzero

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:SRS:CNCodebook:POWer:PZERo
```

class PzeroCls

Pzero commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:SRS:CNCodebook:POWer:PZERo
value: int = driver.configure.signaling.nradio.cell.bwp.srs.cnCodebook.power.
↪pzero.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Sets the SRS power control parameter 'p0' for periodic SRS, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
p_0: No help available

set(*cell_name: str*, *p_0: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:SRS:CNCodebook:POWer:PZERo
driver.configure.signaling.nradio.cell.bwp.srs.cnCodebook.power.pzero.set(cell_
↪name = 'abc', p_0 = 1, bwParts = repcap.BwParts.Default)
```

Sets the SRS power control parameter 'p0' for periodic SRS, for BWP <bb>.

param cell_name
No help available

param p_0
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.167 Resource

class ResourceCls

Resource commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.srs.cnCodebook.resource.clone()
```

Subgroups

6.3.4.10.2.168 Fhopping

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:SRS:CNCodebook:RESource:FHOPping
```

class FhoppingCls

Fhopping commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Csrs: int: No parameter help available
- Bsrs: int: No parameter help available
- Bhop: int: No parameter help available

get(*cell_name: str*, *resource_no: int*, *bwParts=BwParts.Default*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:SRS:CNCodebook:RESource:FHOPping
value: GetStruct = driver.configure.signaling.nradio.cell.bwp.srs.cnCodebook.
↪resource.fhopping.get(cell_name = 'abc', resource_no = 1, bwParts = repcap.
↪BwParts.Default)
```

Configures the frequency hopping for SRS resource <ResourceNo> for periodic SRS, for BWP <bb>.

param cell_name
No help available

param resource_no
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *resource_no: int*, *csrs: int*, *bsrs: int = None*, *bhop: int = None*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:SRS:CNCodebook:RESource:FHOPping
driver.configure.signaling.nradio.cell.bwp.srs.cnCodebook.resource.fhopping.
↪set(cell_name = 'abc', resource_no = 1, csrs = 1, bsrs = 1, bhop = 1, bwParts␣
↪= repcap.BwParts.Default)
```

Configures the frequency hopping for SRS resource <ResourceNo> for periodic SRS, for BWP <bb>.

param cell_name
No help available

param resource_no
No help available

param csrs
No help available

param bsrs
No help available

param bhop
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.169 Resource

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:SRS:CNCodebook:RESource:RESource
```

class ResourceCls

Resource commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Resource_Id: int: No parameter help available
- No_Srs_Ports: enums.AntNoPorts: Number of antenna ports for SRS transmissions
- Fd_Position: int: Frequency domain position
- Fd_Shift: int: Frequency domain shift
- Sequence_Id: int: No parameter help available

class SetStruct

Structure for setting input parameters. Contains optional setting parameters. Fields:

- Cell_Name: str: No parameter help available
- Resource_No: int: No parameter help available
- Resource_Id: int: No parameter help available
- No_Srs_Ports: enums.AntNoPorts: Optional setting parameter. Number of antenna ports for SRS transmissions
- Fd_Position: int: Optional setting parameter. Frequency domain position
- Fd_Shift: int: Optional setting parameter. Frequency domain shift
- Sequence_Id: int: No parameter help available

get(*cell_name: str*, *resource_no: int*, *bwParts=BwParts.Default*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:SRS:CNCodebook:RESource:RESource
value: GetStruct = driver.configure.signaling.nradio.cell.bwp.srs.cnCodebook.
↪resource.resource.get(cell_name = 'abc', resource_no = 1, bwParts = repcap.
↪BwParts.Default)
```

Configures the SRS resource <ResourceNo> for periodic SRS, for BWP <bb>.

param cell_name
No help available

param resource_no
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*structure: SetStruct*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:SRS:CNCodebook:RESource:RESource
structure = driver.configure.signaling.nradio.cell.bwp.srs.cnCodebook.resource.
↪resource.SetStruct()
structure.Cell_Name: str = 'abc'
structure.Resource_No: int = 1
structure.Resource_Id: int = 1
structure.No_Srs_Ports: enums.AntNoPorts = enums.AntNoPorts.P1
structure.Fd_Position: int = 1
structure.Fd_Shift: int = 1
structure.Sequence_Id: int = 1
driver.configure.signaling.nradio.cell.bwp.srs.cnCodebook.resource.resource.
↪set(structure, bwParts = repcap.BwParts.Default)
```

Configures the SRS resource <ResourceNo> for periodic SRS, for BWP <bb>.

param structure
for set value, see the help for SetStruct structure arguments.

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.170 Rmapping

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:SRS:CNCodebook:RESource:RMAPping
```

class RmappingCls

Rmapping commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Start_Position: int: No parameter help available
- No_Symbols: enums.NoSymbolsN: No parameter help available
- Rep_Factor: enums.NoSymbolsN: Repetition factor

get(*cell_name: str*, *resource_no: int*, *bwParts=BwParts.Default*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:SRS:CNCodebook:RESource:RMAPping
value: GetStruct = driver.configure.signaling.nradio.cell.bwp.srs.cnCodebook.
↪resource.rmapping.get(cell_name = 'abc', resource_no = 1, bwParts = repcap.
↪BwParts.Default)
```

Configures the resource mapping for SRS transmissions, for SRS resource <ResourceNo> for periodic SRS, for BWP <bb>.

param cell_name
No help available

param resource_no
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *resource_no: int*, *start_position: int*, *no_symbols: NoSymbolsN = None*, *rep_factor: NoSymbolsN = None*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:SRS:CNCodebook:RESource:RMAPping
driver.configure.signaling.nradio.cell.bwp.srs.cnCodebook.resource.rmapping.
↪set(cell_name = 'abc', resource_no = 1, start_position = 1, no_symbols =␣
↪enums.NoSymbolsN.N1, rep_factor = enums.NoSymbolsN.N1, bwParts = repcap.
↪BwParts.Default)
```

Configures the resource mapping for SRS transmissions, for SRS resource <ResourceNo> for periodic SRS, for BWP <bb>.

param cell_name
No help available

param resource_no
No help available

param start_position
No help available

param no_symbols
No help available

param rep_factor
Repetition factor

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.171 Rtype

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:SRS:CNCodebook:RESource:RTYPe
```

`class RtypeCls`

Rtype commands group definition. 1 total commands, 0 Subgroups, 1 group commands

`class GetStruct`

Response structure. Fields:

- Td_Type: enums.TdType: APERiodic: no SRS transmissions PERiodic: SRS transmissions in every nth slot
- Period: int: Periodicity of slots (SRS every nth slot)
- Offset: int: Offset as number of slots. Must be smaller than the Period.

`get`(*cell_name: str*, *resource_no: int*, *bwParts=BwParts.Default*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:SRS:CNCodebook:RESource:RTYPe
value: GetStruct = driver.configure.signaling.nradio.cell.bwp.srs.cnCodebook.
↪resource.rtype.get(cell_name = 'abc', resource_no = 1, bwParts = repcap.
↪BwParts.Default)
```

Configures the resource type for SRS resource <ResourceNo> for periodic SRS, for BWP <bb>.

param cell_name
No help available

param resource_no
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *resource_no: int*, *td_type: TdType*, *period: int = None*, *offset: int = None*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:SRS:CNCodebook:RESource:RTYPe
driver.configure.signaling.nradio.cell.bwp.srs.cnCodebook.resource.rtype.
↪set(cell_name = 'abc', resource_no = 1, td_type = enums.TdType.APERiodic,␣
↪period = 1, offset = 1, bwParts = repcap.BwParts.Default)
```

Configures the resource type for SRS resource <ResourceNo> for periodic SRS, for BWP <bb>.

param cell_name
No help available

param resource_no
No help available

param td_type
APERiodic: no SRS transmissions PERiodic: SRS transmissions in every nth slot

param period
Periodicity of slots (SRS every nth slot)

param offset
Offset as number of slots. Must be smaller than the Period.

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.172 Tcomb

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:SRS:CNCodebook:RESource:TCOMb
```

class TcombCls

Tcomb commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Ktc: enums.Ktc: No parameter help available
- Offset: int: No parameter help available
- Cyclic_Shift: int: No parameter help available

get(*cell_name: str*, *resource_no: int*, *bwParts=BwParts.Default*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:SRS:CNCodebook:RESource:TCOMb
value: GetStruct = driver.configure.signaling.nradio.cell.bwp.srs.cnCodebook.
↪resource.tcomb.get(cell_name = 'abc', resource_no = 1, bwParts = repcap.
↪BwParts.Default)
```

Configures the comb structure of the SRS resource <ResourceNo> for periodic SRS, for BWP <bb>.

param cell_name
No help available

param resource_no
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *resource_no: int*, *ktc: Ktc = None*, *offset: int = None*, *cyclic_shift: int = None*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:SRS:CNCodebook:RESource:TCOMb
driver.configure.signaling.nradio.cell.bwp.srs.cnCodebook.resource.tcomb.
↪set(cell_name = 'abc', resource_no = 1, ktc = enums.Ktc.N2, offset = 1,␣
↪cyclic_shift = 1, bwParts = repcap.BwParts.Default)
```

Configures the comb structure of the SRS resource <ResourceNo> for periodic SRS, for BWP <bb>.

param cell_name
No help available

param resource_no
No help available

param ktc
No help available

param offset
No help available

param cyclic_shift
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.173 Scheduler

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:SRS:CNCodebook:SCHeduler
```

class SchedulerCls

Scheduler commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Ncoherent_Tpmi: enums.NcoherentTpmi: No parameter help available
- Tpmi_Layers: enums.MaxLength: No parameter help available
- Tpmi: enums.Tpmi: No parameter help available
- Resource_Id: enums.ResourceId: No parameter help available

get(*cell_name: str*, *bwParts=BwParts.Default*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:SRS:CNCodebook:SCHeduler
value: GetStruct = driver.configure.signaling.nradio.cell.bwp.srs.cnCodebook.
↪scheduler.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

No command help available

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *ncoherent_tpmi: NcoherentTpmi*, *tpmi_layers: MaxLength = None*, *tpmi: Tpmi = None*, *resource_id: ResourceId = None*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:SRS:CNCodebook:SCHeduler
driver.configure.signaling.nradio.cell.bwp.srs.cnCodebook.scheduler.set(cell_
↪name = 'abc', ncoherent_tpmi = enums.NcoherentTpmi.FPARtial, tpmi_layers =␣
↪enums.MaxLength.L1, tpmi = enums.Tpmi.T0, resource_id = enums.ResourceId.R1,␣
↪bwParts = repcap.BwParts.Default)
```

No command help available

param cell_name
No help available

param ncoherent_tpmi
No help available

param tpmi_layers
No help available

param tpmi
No help available

param resource_id
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.174 TdBehavior

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:SRS:CNCodebook:TDBehavior
```

class TdBehaviorCls

TdBehavior commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → TdType

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:SRS:CNCodebook:TDBehavior
value: enums.TdType = driver.configure.signaling.nradio.cell.bwp.srs.cnCodebook.
↪tdBehavior.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Selects the time domain behavior ('resourceType') of the SRS resource set and thus enables or disables periodic SRS, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
td_behavior: APERiodic: no SRS transmissions PERiodic: SRS transmissions in every nth slot

set(*cell_name: str*, *td_behavior: TdType*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:SRS:CNCodebook:TDBehavior
driver.configure.signaling.nradio.cell.bwp.srs.cnCodebook.tdBehavior.set(cell_
↪name = 'abc', td_behavior = enums.TdType.APERiodic, bwParts = repcap.BwParts.
↪Default)
```

Selects the time domain behavior ('resourceType') of the SRS resource set and thus enables or disables periodic SRS, for BWP <bb>.

param cell_name
No help available

param td_behavior
APERiodic: no SRS transmissions PERiodic: SRS transmissions in every nth slot

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.175 Usage

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:SRS:CNCodebook:USAGe
```

class UsageCls

Usage commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → Schema

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:SRS:CNCodebook:USAGe
value: enums.Schema = driver.configure.signaling.nradio.cell.bwp.srs.cnCodebook.
↪usage.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Selects the usage of the SRS resource set for periodic SRS, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
schema: Codebook, non-codebook

set(*cell_name: str*, *schema: Schema*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:SRS:CNCodebook:USAGe
driver.configure.signaling.nradio.cell.bwp.srs.cnCodebook.usage.set(cell_name =
↪'abc', schema = enums.Schema.CODebook, bwParts = repcap.BwParts.Default)
```

Selects the usage of the SRS resource set for periodic SRS, for BWP <bb>.

param cell_name
No help available

param schema
Codebook, non-codebook

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.176 Sspacing

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwpid>:SSPacing
```

class SspacingCls

Sspacing commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwpid>:SSPacing
value: int = driver.configure.signaling.nradio.cell.bwp.sspacing.get(cell_name␣
↪= 'abc', bwParts = repcap.BwParts.Default)
```

Configures the subcarrier spacing for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
spacing: No help available

set(*cell_name: str*, *spacing: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwpid>:SSPacing
driver.configure.signaling.nradio.cell.bwp.sspacing.set(cell_name = 'abc',␣
↪spacing = 1, bwParts = repcap.BwParts.Default)
```

Configures the subcarrier spacing for BWP <bb>.

param cell_name
No help available

param spacing
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.177 Sue

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwpid>:SUE
```

class SueCls

Sue commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwpid>:SUE
value: bool = driver.configure.signaling.nradio.cell.bwp.sue.get(cell_name =
↪'abc', bwParts = repcap.BwParts.Default)
```

Enables sending information about BWP <bb> to the UE.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
enable: No help available

set(*cell_name: str*, *enable: bool*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwpid>:SUE
driver.configure.signaling.nradio.cell.bwp.sue.set(cell_name = 'abc', enable =␣
↪False, bwParts = repcap.BwParts.Default)
```

Enables sending information about BWP <bb> to the UE.

param cell_name
No help available

param enable
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.178 Tadvance

class TadvanceCls

Tadvance commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.tadvance.clone()
```

Subgroups

6.3.4.10.2.179 Periodicity

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:TADVance:PERiodicity
```

class PeriodicityCls

Periodicity commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → TadvPeriodicity

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:TADVance:PERiodicity
value: enums.TadvPeriodicity = driver.configure.signaling.nradio.cell.bwp.
↪tadvance.periodicity.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Configures the periodicity for sending timing advance commands to the UE, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
periodicity: No help available

set(*cell_name: str*, *periodicity: TadvPeriodicity*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:TADVance:PERiodicity
driver.configure.signaling.nradio.cell.bwp.tadvance.periodicity.set(cell_name =
↪'abc', periodicity = enums.TadvPeriodicity.CONTinuous, bwParts = repcap.
↪BwParts.Default)
```

Configures the periodicity for sending timing advance commands to the UE, for BWP <bb>.

param cell_name
No help available

param periodicity
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.180 Timing

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:TADVance:TIMing
```

class TimingCls

Timing commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:TADVance:TIMing
value: int = driver.configure.signaling.nradio.cell.bwp.tadvance.timing.
↪get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Configures a timing advance value to be sent to the UE, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
ta: No help available

set(*cell_name: str*, *ta: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:TADVance:TIMing
driver.configure.signaling.nradio.cell.bwp.tadvance.timing.set(cell_name = 'abc
↪', ta = 1, bwParts = repcap.BwParts.Default)
```

Configures a timing advance value to be sent to the UE, for BWP <bb>.

param cell_name
No help available

param ta
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.181 Target

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP:TARGet
```

class TargetCls

Target commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Active_Dl_Bwp: int: Selects the target DL BWP. IBWP: initial BWP integer: BWP ID
- Active_Ul_Bwp: int: Selects the target UL BWP. IBWP: initial BWP integer: BWP ID SADL: same as DL

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP:TARGet
value: GetStruct = driver.configure.signaling.nradio.cell.bwp.target.get(cell_
↪name = 'abc')
```

Switches the active BWP.

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *active_dl_bwp: int*, *active_ul_bwp: int = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP:TARGet
driver.configure.signaling.nradio.cell.bwp.target.set(cell_name = 'abc', active_
↪dl_bwp = 1, active_ul_bwp = 1)
```

Switches the active BWP.

param cell_name
No help available

param active_dl_bwp
Selects the target DL BWP. IBWP: initial BWP integer: BWP ID

param active_ul_bwp
Selects the target UL BWP. IBWP: initial BWP integer: BWP ID SADL: same as DL

6.3.4.10.2.182 UeScheduling

class UeSchedulingCls

UeScheduling commands group definition. 75 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.ueScheduling.clone()
```

Subgroups

6.3.4.10.2.183 CmMapping

class CmMappingCls

CmMapping commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.ueScheduling.cmMapping.clone()
```

Subgroups

6.3.4.10.2.184 Mcs

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:CMMapping:MCS
```

class McsCls

Mcs commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → List[int]

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:CMMapping:MCS
value: List[int] = driver.configure.signaling.nradio.cell.bwp.ueScheduling.
↪cmMapping.mcs.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Sets the configuration mode to UDEFined and defines the mapping table for that mode, for BWP <bb>. A query returns the mapping table contents for the currently used configuration mode, without changing the mode. For setting the configuration mode, see [CONFigure:]SIGNaling:NRADio:CELL:UESCheduling:CMMapping:MCSTable.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
mcs: Comma-separated list of 16 MCS values, for reported CQI values 0 to 15.

set(*cell_name: str*, *mcs: List[int]*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:CMMapping:MCS
driver.configure.signaling.nradio.cell.bwp.ueScheduling.cmMapping.mcs.set(cell_
↪name = 'abc', mcs = [1, 2, 3], bwParts = repcap.BwParts.Default)
```

Sets the configuration mode to UDEFined and defines the mapping table for that mode, for BWP <bb>. A query returns the mapping table contents for the currently used configuration mode, without changing the mode. For setting the configuration mode, see [CONFigure:]SIGNaling:NRADio:CELL:UESCheduling:CMMapping:MCSTable.

param cell_name
No help available

param mcs
Comma-separated list of 16 MCS values, for reported CQI values 0 to 15.

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.185 McsTable

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:CMMapping:MCSTable
```

class McsTableCls

McsTable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- **Mcs_Table: enums.McsTableC:**
  - AUTO: The mapping table is selected automatically, depending on [CONFigure:]SIGNaling:NRADio:CELL:UESCheduling:UDEFined:DL:MCSTable.
  - P521: The mapping table contents are defined by 3GPP TS 38.521-4.Table selection via Predefined3GPP.
  - UDEFined: The mapping table contents are defined via a separate command, see [CONFigure:]SIGNaling:NRADio:CELL:UESCheduling:CMMapping:MCS.

- Predefined_3_Gpp: enums.ConfigTypeB: Selects a mapping table for MCSTable = P521. T1: table A.4-1 in 3GPP TS 38.

521-4 T2: table A.4-2 in 3GPP TS 38.521-4 T3: table A.4-3 in 3GPP TS 38.521-4

get(*cell_name: str*, *bwParts=BwParts.Default*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:CMMapping:MCSTable
value: GetStruct = driver.configure.signaling.nradio.cell.bwp.ueScheduling.
↪cmMapping.mcsTable.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Selects a configuration mode for the CQI-MCS mapping table for follow WB CQI, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *mcs_table: McsTableC*, *predefined_3_gpp: ConfigTypeB = None*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:CMMapping:MCSTable
driver.configure.signaling.nradio.cell.bwp.ueScheduling.cmMapping.mcsTable.
↪set(cell_name = 'abc', mcs_table = enums.McsTableC.AUTO, predefined_3_gpp =␣
↪enums.ConfigTypeB.T1, bwParts = repcap.BwParts.Default)
```

Selects a configuration mode for the CQI-MCS mapping table for follow WB CQI, for BWP <bb>.

param cell_name
No help available

param mcs_table

- AUTO: The mapping table is selected automatically, depending on [CONFigure:]SIGNaling:NRADio:CELL:UESCheduling:UDEFined:DL:MCSTable.
- P521: The mapping table contents are defined by 3GPP TS 38.521-4.Table selection via Predefined3GPP.
- UDEFined: The mapping table contents are defined via a separate command, see [CONFigure:]SIGNaling:NRADio:CELL:UESCheduling:CMMapping:MCS.

param predefined_3_gpp
Selects a mapping table for MCSTable = P521. T1: table A.4-1 in 3GPP TS 38.521-4 T2: table A.

4-2 in 3GPP TS 38.521-4 T3: table A.4-3 in 3GPP TS 38.521-4 :param bwParts: optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.186 Smode

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SMODe
```

class SmodeCls

Smode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → ModeUeScheduling

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SMODe
value: enums.ModeUeScheduling = driver.configure.signaling.nradio.cell.bwp.
↪ueScheduling.smode.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Selects a scheduling mode for DL and UL, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
mode: FIXed: Fixed scheduling, DL and UL. SPS: SPS DL, CG UL. CQI: Follow CQI WB DL, fixed scheduling UL. PRI: Follow PMI WB + RI DL, fixed scheduling UL. CPRI: Follow CQI WB + PMI WB + RI DL, fixed scheduling UL. BO: Follow buffer occupancy (BO) DL, fixed scheduling UL. UDEFined: Other dynamic scheduling mode (query only) .

set(*cell_name: str*, *mode: ModeUeScheduling*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SMODe
driver.configure.signaling.nradio.cell.bwp.ueScheduling.smode.set(cell_name =
↪'abc', mode = enums.ModeUeScheduling.BO, bwParts = repcap.BwParts.Default)
```

Selects a scheduling mode for DL and UL, for BWP <bb>.

param cell_name
No help available

param mode
FIXed: Fixed scheduling, DL and UL. SPS: SPS DL, CG UL. CQI: Follow CQI WB DL, fixed scheduling UL. PRI: Follow PMI WB + RI DL, fixed scheduling UL. CPRI: Follow CQI WB + PMI WB + RI DL, fixed scheduling UL. BO: Follow buffer occupancy (BO) DL, fixed scheduling UL. UDEFined: Other dynamic scheduling mode (query only) .

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.187 Sps

class SpsCls

Sps commands group definition. 36 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.clone()
```

Subgroups

6.3.4.10.2.188 Downlink

class DownlinkCls

Downlink commands group definition. 10 total commands, 10 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.downlink.clone()
```

Subgroups

6.3.4.10.2.189 Alevel

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:DL:ALEVel
```

class AlevelCls

Alevel commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → Level

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:DL:ALEVel
value: enums.Level = driver.configure.signaling.nradio.cell.bwp.ueScheduling.
↪sps.downlink.alevel.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Configures the aggregation level for DL SPS scheduling, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
alevel: No help available

set(*cell_name: str*, *alevel: Level*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:DL:ALEVel
driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.downlink.alevel.
↪set(cell_name = 'abc', alevel = enums.Level.AL1, bwParts = repcap.BwParts.
↪Default)
```

Configures the aggregation level for DL SPS scheduling, for BWP <bb>.

param cell_name
No help available

param alevel
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.190 All

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:DL:ALL
```

class AllCls

All commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Periodicity: int: No parameter help available
- Mcs_Table: enums.McsTableB: 256QAM, 64QAM low SE, 64QAM
- Alevel: enums.Level: Aggregation level
- Search_Space_Id: int: No parameter help available
- Resource_Allocation_Type: enums.ResourceAllocationType: No parameter help available
- Rgb_Size: enums.RgbSize: Config 1 or 2
- No_Harq: int: Signaled 'nrofHARQ-Processes'
- Mapping: enums.MappingI: Interleaved or non-interleaved virtual RB to physical RB mapping
- Padding: enums.SpsPadding: No DL padding or with DL padding

class SetStruct

Structure for setting input parameters. Contains optional setting parameters. Fields:

- Cell_Name: str: No parameter help available
- Periodicity: int: No parameter help available
- Mcs_Table: enums.McsTableB: Optional setting parameter. 256QAM, 64QAM low SE, 64QAM
- Alevel: enums.Level: Optional setting parameter. Aggregation level
- Search_Space_Id: int: No parameter help available

- Resource_Allocation_Type: enums.ResourceAllocationType: No parameter help available
- Rgb_Size: enums.RgbSize: Optional setting parameter. Config 1 or 2
- No_Harq: int: Optional setting parameter. Signaled 'nrofHARQ-Processes'
- Mapping: enums.MappingI: Optional setting parameter. Interleaved or non-interleaved virtual RB to physical RB mapping
- Padding: enums.SpsPadding: Optional setting parameter. No DL padding or with DL padding

get(*cell_name: str*, *bwParts=BwParts.Default*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:DL:ALL
value: GetStruct = driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.
↪downlink.all.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Configures several settings for SPS DL scheduling (combination of the other ...SPS:DL:... commands) , for BWP <bb>.

param cell_name
Type 0, type 1, dynamic switch

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*structure: SetStruct*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:DL:ALL
structure = driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.
↪downlink.all.SetStruct()
structure.Cell_Name: str = 'abc'
structure.Periodicity: int = 1
structure.Mcs_Table: enums.McsTableB = enums.McsTableB.L64
structure.Alevel: enums.Level = enums.Level.AL1
structure.Search_Space_Id: int = 1
structure.Resource_Allocation_Type: enums.ResourceAllocationType = enums.
↪ResourceAllocationType.DSWich
structure.Rgb_Size: enums.RgbSize = enums.RgbSize.CON1
structure.No_Harq: int = 1
structure.Mapping: enums.MappingI = enums.MappingI.INT
structure.Padding: enums.SpsPadding = enums.SpsPadding.ALLZero
driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.downlink.all.
↪set(structure, bwParts = repcap.BwParts.Default)
```

Configures several settings for SPS DL scheduling (combination of the other ...SPS:DL:... commands) , for BWP <bb>.

param structure
for set value, see the help for SetStruct structure arguments.

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.191 Mapping

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:DL:MAPPing
```

class MappingCls

Mapping commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → MappingI

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:DL:MAPPing
value: enums.MappingI = driver.configure.signaling.nradio.cell.bwp.ueScheduling.
↪sps.downlink.mapping.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Selects whether interleaved or non-interleaved virtual RB to physical RB mapping is applied for the PDSCH, for DL SPS scheduling, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
mapping: Interleaved or non-interleaved

set(*cell_name: str*, *mapping: MappingI*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:DL:MAPPing
driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.downlink.mapping.
↪set(cell_name = 'abc', mapping = enums.MappingI.INT, bwParts = repcap.BwParts.
↪Default)
```

Selects whether interleaved or non-interleaved virtual RB to physical RB mapping is applied for the PDSCH, for DL SPS scheduling, for BWP <bb>.

param cell_name
No help available

param mapping
Interleaved or non-interleaved

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.192 McsTable

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:DL:MCSTable
```

class McsTableCls

McsTable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → McsTableB

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:DL:MCSTable
value: enums.McsTableB = driver.configure.signaling.nradio.cell.bwp.
↪ueScheduling.sps.downlink.mcsTable.get(cell_name = 'abc', bwParts = repcap.
↪BwParts.Default)
```

Configures the mcs-Table signaled for DL SPS scheduling, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
mcs_table: 256QAM, 64QAM low SE, 64QAM

set(*cell_name: str*, *mcs_table: McsTableB*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:DL:MCSTable
driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.downlink.mcsTable.
↪set(cell_name = 'abc', mcs_table = enums.McsTableB.L64, bwParts = repcap.
↪BwParts.Default)
```

Configures the mcs-Table signaled for DL SPS scheduling, for BWP <bb>.

param cell_name
No help available

param mcs_table
256QAM, 64QAM low SE, 64QAM

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.193 Nohp

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:DL:NOHP
```

class NohpCls

Nohp commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:DL:NOHP
value: int = driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.
↪downlink.nohp.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Configures the signaled 'nrofHARQ-Processes' for DL SPS scheduling, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
no_harq: No help available

set(*cell_name: str*, *no_harq: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:DL:NOHP
driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.downlink.nohp.
↪set(cell_name = 'abc', no_harq = 1, bwParts = repcap.BwParts.Default)
```

Configures the signaled 'nrofHARQ-Processes' for DL SPS scheduling, for BWP <bb>.

param cell_name
No help available

param no_harq
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.194 Padding

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:DL:PADDing
```

class PaddingCls

Padding commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → SpsPadding

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:DL:PADDing
value: enums.SpsPadding = driver.configure.signaling.nradio.cell.bwp.
↪ueScheduling.sps.downlink.padding.get(cell_name = 'abc', bwParts = repcap.
↪BwParts.Default)
```

Activates or deactivates downlink padding for SPS scheduling, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
padding: No help available

set(*cell_name: str*, *padding: SpsPadding*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:DL:PADDing
driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.downlink.padding.
↪set(cell_name = 'abc', padding = enums.SpsPadding.ALLZero, bwParts = repcap.
↪BwParts.Default)
```

Activates or deactivates downlink padding for SPS scheduling, for BWP <bb>.

param cell_name
No help available

param padding
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.195 Periodicity

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:DL:PERiodicity
```

class PeriodicityCls

Periodicity commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:DL:PERiodicity
value: int = driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.
↪downlink.periodicity.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Configures the signaled 'periodicity' for DL SPS scheduling, in ms, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
periodicity: No help available

set(*cell_name: str*, *periodicity: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:DL:PERiodicity
driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.downlink.
↪periodicity.set(cell_name = 'abc', periodicity = 1, bwParts = repcap.BwParts.
↪Default)
```

Configures the signaled 'periodicity' for DL SPS scheduling, in ms, for BWP <bb>.

param cell_name
No help available

param periodicity
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.196 RaType

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:DL:RATYpe
```

class RaTypeCls

RaType commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → ResourceAllocationType

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:DL:RATYpe
value: enums.ResourceAllocationType = driver.configure.signaling.nradio.cell.
↪bwp.ueScheduling.sps.downlink.raType.get(cell_name = 'abc', bwParts = repcap.
↪BwParts.Default)
```

Configures the signaled 'resourceAllocation' for DL SPS scheduling, for BWP <bb>.

param cell_name
type 0, type 1, dynamic switch

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
resource_allocation_type: No help available

set(*cell_name: str*, *resource_allocation_type: ResourceAllocationType*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:DL:RATYpe
driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.downlink.raType.
↪set(cell_name = 'abc', resource_allocation_type = enums.
↪ResourceAllocationType.DSWich, bwParts = repcap.BwParts.Default)
```

Configures the signaled 'resourceAllocation' for DL SPS scheduling, for BWP <bb>.

param cell_name
No help available

param resource_allocation_type
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.197 RbgSize

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:DL:RBGSize
```

class RbgSizeCls

RbgSize commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → RgbSize

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:DL:RBGSize
value: enums.RgbSize = driver.configure.signaling.nradio.cell.bwp.ueScheduling.
↪sps.downlink.rbgSize.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Configures the signaled 'rbg-Size' for DL SPS scheduling, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
rgb_size: Config 1 or 2

set(*cell_name: str*, *rgb_size: RgbSize*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:DL:RBGSize
driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.downlink.rbgSize.
↪set(cell_name = 'abc', rgb_size = enums.RgbSize.CON1, bwParts = repcap.
↪BwParts.Default)
```

Configures the signaled 'rbg-Size' for DL SPS scheduling, for BWP <bb>.

param cell_name
No help available

param rgb_size
Config 1 or 2

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.198 Ssid

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:DL:SSID
```

class SsidCls

Ssid commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:DL:SSID
value: int = driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.
↪downlink.ssid.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Configures the search space ID for DL SPS scheduling, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
search_space_id: No help available

set(*cell_name: str*, *search_space_id: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:DL:SSID
driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.downlink.ssid.
↪set(cell_name = 'abc', search_space_id = 1, bwParts = repcap.BwParts.Default)
```

Configures the search space ID for DL SPS scheduling, for BWP <bb>.

param cell_name
No help available

param search_space_id
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.199 Sassignment

class SassignmentCls

Sassignment commands group definition. 14 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.sassignment.clone()
```

Subgroups

6.3.4.10.2.200 Downlink

class DownlinkCls

Downlink commands group definition. 7 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.sassignment.
↪downlink.clone()
```

Subgroups

6.3.4.10.2.201 All

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:SASSignment:DL:ALL
```

class AllCls

All commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Slot: int: Slot for sending the DCI that activates DL SPS.
- Number_Rb: int: No parameter help available
- Start_Rb: int: No parameter help available
- Mcs: int: No parameter help available
- Start_Symbol: int: No parameter help available
- Number_Symbol: int: No parameter help available
- Mapping: enums.Mapping: Type of PDSCH mapping
- Offset: int: Slot offset k0 for the PDSCH

class SetStruct

Structure for setting input parameters. Contains optional setting parameters. Fields:

- Cell_Name: str: No parameter help available
- Slot: int: Optional setting parameter. Slot for sending the DCI that activates DL SPS.
- Number_Rb: int: No parameter help available
- Start_Rb: int: No parameter help available
- Mcs: int: No parameter help available
- Start_Symbol: int: No parameter help available
- Number_Symbol: int: No parameter help available
- Mapping: enums.Mapping: Optional setting parameter. Type of PDSCH mapping
- Offset: int: Optional setting parameter. Slot offset k0 for the PDSCH

get(*cell_name: str*, *bwParts=BwParts.Default*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:SASSignment:DL:ALL
value: GetStruct = driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.
↪sassignment.downlink.all.get(cell_name = 'abc', bwParts = repcap.BwParts.
↪Default)
```

Configures several settings for SPS DL scheduling (combination of the other ...SPS:SASSignment:DL:... commands) , for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*structure: SetStruct*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:SASSignment:DL:ALL
structure = driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.
↪sassignment.downlink.all.SetStruct()
structure.Cell_Name: str = 'abc'
structure.Slot: int = 1
structure.Number_Rb: int = 1
structure.Start_Rb: int = 1
structure.Mcs: int = 1
structure.Start_Symbol: int = 1
structure.Number_Symbol: int = 1
structure.Mapping: enums.Mapping = enums.Mapping.A
structure.Offset: int = 1
driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.sassignment.
↪downlink.all.set(structure, bwParts = repcap.BwParts.Default)
```

Configures several settings for SPS DL scheduling (combination of the other ...SPS:SASSignment:DL:... commands) , for BWP <bb>.

param structure
for set value, see the help for SetStruct structure arguments.

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.202 Mcs

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:SASSignment:DL:MCS
```

class McsCls

Mcs commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:SASSignment:DL:MCS
value: int = driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.
↪sassignment.downlink.mcs.get(cell_name = 'abc', bwParts = repcap.BwParts.
↪Default)
```

Specifies the MCS index, for DL SPS scheduling, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
mcs: No help available

set(*cell_name: str*, *mcs: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:SASSignment:DL:MCS
driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.sassignment.
↪downlink.mcs.set(cell_name = 'abc', mcs = 1, bwParts = repcap.BwParts.Default)
```

Specifies the MCS index, for DL SPS scheduling, for BWP <bb>.

param cell_name
No help available

param mcs
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.203 Rb

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:SASSignment:DL:RB
```

class RbCls

Rb commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Number_Rb: int: No parameter help available
- Start_Rb: int: No parameter help available

get(*cell_name: str*, *bwParts=BwParts.Default*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:SASSignment:DL:RB
value: GetStruct = driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.
↪sassignment.downlink.rb.get(cell_name = 'abc', bwParts = repcap.BwParts.
↪Default)
```

Defines the scheduled contiguous RB allocation, within the BWP, for DL SPS scheduling, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *number_rb: int*, *start_rb: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:SASSignment:DL:RB
driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.sassignment.
↪downlink.rb.set(cell_name = 'abc', number_rb = 1, start_rb = 1, bwParts =␣
↪repcap.BwParts.Default)
```

Defines the scheduled contiguous RB allocation, within the BWP, for DL SPS scheduling, for BWP <bb>.

param cell_name
No help available

param number_rb
No help available

param start_rb
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.204 Sindex

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:SASSignment:DL:SINDex
```

class SindexCls

Sindex commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:SASSignment:DL:SINDex
value: int = driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.
↪sassignment.downlink.sindex.get(cell_name = 'abc', bwParts = repcap.BwParts.
↪Default)
```

Selects a slot for sending the DCI that activates DL SPS and informs the UE about the scheduling, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
slot: No help available

set(*cell_name: str*, *slot: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:SASSignment:DL:SINDex
driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.sassignment.
↪downlink.sindex.set(cell_name = 'abc', slot = 1, bwParts = repcap.BwParts.
↪Default)
```

Selects a slot for sending the DCI that activates DL SPS and informs the UE about the scheduling, for BWP <bb>.

param cell_name
No help available

param slot
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.205 Tdomain

class TdomainCls

Tdomain commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.sassignment.
↪downlink.tdomain.clone()
```

Subgroups

6.3.4.10.2.206 Chmapping

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:SASSignment:DL:TDOMain:CHMapping
```

class ChmappingCls

Chmapping commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → Mapping

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:SASSignment:DL:TDOMain:CHMapping
value: enums.Mapping = driver.configure.signaling.nradio.cell.bwp.ueScheduling.
↪sps.sassignment.downlink.tdomain.chmapping.get(cell_name = 'abc', bwParts =␣
↪repcap.BwParts.Default)
```

Selects the type of PDSCH mapping, for DL SPS scheduling, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
mapping: No help available

set(*cell_name: str*, *mapping: Mapping*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:SASSignment:DL:TDOMain:CHMapping
driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.sassignment.
↪downlink.tdomain.chmapping.set(cell_name = 'abc', mapping = enums.Mapping.A,␣
↪bwParts = repcap.BwParts.Default)
```

Selects the type of PDSCH mapping, for DL SPS scheduling, for BWP <bb>.

param cell_name
No help available

param mapping
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.207 Soffset

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:SASSignment:DL:TDOMain:SOFFset
```

class SoffsetCls

Soffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:SASSignment:DL:TDOMain:SOFFset
value: int = driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.
↪sassignment.downlink.tdomain.soffset.get(cell_name = 'abc', bwParts = repcap.
↪BwParts.Default)
```

Configures the slot offset k0 for the PDSCH, for DL SPS scheduling, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
offset: No help available

set(*cell_name: str*, *offset: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:SASSignment:DL:TDOMain:SOFFset
driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.sassignment.
↪downlink.tdomain.soffset.set(cell_name = 'abc', offset = 1, bwParts = repcap.
↪BwParts.Default)
```

Configures the slot offset k0 for the PDSCH, for DL SPS scheduling, for BWP <bb>.

param cell_name
No help available

param offset
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.208 Symbol

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:SASSignment:DL:TDOMain:SYMBol
```

class SymbolCls

Symbol commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Start_Symbol: int: No parameter help available
- Number_Symbol: int: No parameter help available

get(*cell_name: str*, *bwParts=BwParts.Default*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:SASSignment:DL:TDOMain:SYMBol
value: GetStruct = driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.
↪sassignment.downlink.tdomain.symbol.get(cell_name = 'abc', bwParts = repcap.
↪BwParts.Default)
```

Defines the index of the first allocated OFDM symbol and the number of allocated OFDM symbols, for DL SPS scheduling, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *start_symbol: int*, *number_symbol: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:SASSignment:DL:TDOMain:SYMBol
driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.sassignment.
↪downlink.tdomain.symbol.set(cell_name = 'abc', start_symbol = 1, number_
↪symbol = 1, bwParts = repcap.BwParts.Default)
```

Defines the index of the first allocated OFDM symbol and the number of allocated OFDM symbols, for DL SPS scheduling, for BWP <bb>.

param cell_name
No help available

param start_symbol
No help available

param number_symbol
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.209 Uplink

class UplinkCls

Uplink commands group definition. 7 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.sassignment.uplink.
↪clone()
```

Subgroups

6.3.4.10.2.210 All

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:SASSignment:UL:ALL
```

class AllCls

All commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Slot: int: Slot for sending the DCI that enables UL CG.
- Number_Rb: int: No parameter help available
- Start_Rb: int: No parameter help available
- Mcs: int: No parameter help available
- Start_Symbol: int: No parameter help available
- Number_Symbol: int: No parameter help available
- Mapping: enums.Mapping: Type of PUSCH mapping
- Offset: int: Slot offset k2 for the PUSCH

class SetStruct

Structure for setting input parameters. Contains optional setting parameters. Fields:

- Cell_Name: str: No parameter help available

- Slot: int: Optional setting parameter. Slot for sending the DCI that enables UL CG.
- Number_Rb: int: No parameter help available
- Start_Rb: int: No parameter help available
- Mcs: int: No parameter help available
- Start_Symbol: int: No parameter help available
- Number_Symbol: int: No parameter help available
- Mapping: enums.Mapping: Optional setting parameter. Type of PUSCH mapping
- Offset: int: Optional setting parameter. Slot offset k2 for the PUSCH

get(*cell_name: str*, *bwParts=BwParts.Default*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:SASSignment:UL:ALL
value: GetStruct = driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.
↪sassignment.uplink.all.get(cell_name = 'abc', bwParts = repcap.BwParts.
↪Default)
```

Configures several settings for UL configured grant (combination of the other ...SPS:SASSignment:UL:... commands) , for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*structure: SetStruct*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:SASSignment:UL:ALL
structure = driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.
↪sassignment.uplink.all.SetStruct()
structure.Cell_Name: str = 'abc'
structure.Slot: int = 1
structure.Number_Rb: int = 1
structure.Start_Rb: int = 1
structure.Mcs: int = 1
structure.Start_Symbol: int = 1
structure.Number_Symbol: int = 1
structure.Mapping: enums.Mapping = enums.Mapping.A
structure.Offset: int = 1
driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.sassignment.uplink.
↪all.set(structure, bwParts = repcap.BwParts.Default)
```

Configures several settings for UL configured grant (combination of the other ...SPS:SASSignment:UL:... commands) , for BWP <bb>.

param structure
for set value, see the help for SetStruct structure arguments.

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.211 Mcs

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:SASSignment:UL:MCS
```

class McsCls

Mcs commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:SASSignment:UL:MCS
value: int = driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.
↪sassignment.uplink.mcs.get(cell_name = 'abc', bwParts = repcap.BwParts.
↪Default)
```

Specifies the MCS index, for UL configured grant, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
mcs: No help available

set(*cell_name: str*, *mcs: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:SASSignment:UL:MCS
driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.sassignment.uplink.
↪mcs.set(cell_name = 'abc', mcs = 1, bwParts = repcap.BwParts.Default)
```

Specifies the MCS index, for UL configured grant, for BWP <bb>.

param cell_name
No help available

param mcs
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.212 Rb

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:SASSignment:UL:RB
```

class RbCls

Rb commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Number_Rb: int: No parameter help available
- Start_Rb: int: No parameter help available

get(*cell_name: str*, *bwParts=BwParts.Default*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:SASSignment:UL:RB
value: GetStruct = driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.
↪sassignment.uplink.rb.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Defines the scheduled contiguous RB allocation, within the BWP, for UL configured grant, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *number_rb: int*, *start_rb: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:SASSignment:UL:RB
driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.sassignment.uplink.
↪rb.set(cell_name = 'abc', number_rb = 1, start_rb = 1, bwParts = repcap.
↪BwParts.Default)
```

Defines the scheduled contiguous RB allocation, within the BWP, for UL configured grant, for BWP <bb>.

param cell_name
No help available

param number_rb
No help available

param start_rb
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.213 Sindex

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:SASSignment:UL:SINDex
```

class SindexCls

Sindex commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:SASSignment:UL:SINDex
value: int = driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.
↪sassignment.uplink.sindex.get(cell_name = 'abc', bwParts = repcap.BwParts.
↪Default)
```

Selects a slot for sending the DCI that informs the UE about the UL grant, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
slot: No help available

set(*cell_name: str*, *slot: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:SASSignment:UL:SINDex
driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.sassignment.uplink.
↪sindex.set(cell_name = 'abc', slot = 1, bwParts = repcap.BwParts.Default)
```

Selects a slot for sending the DCI that informs the UE about the UL grant, for BWP <bb>.

param cell_name
No help available

param slot
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.214 Tdomain

class TdomainCls

Tdomain commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.sassignment.uplink.
↪tdomain.clone()
```

Subgroups

6.3.4.10.2.215 Chmapping

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:SASSignment:UL:TDOMain:CHMapping
```

class ChmappingCls

Chmapping commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → Mapping

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:SASSignment:UL:TDOMain:CHMapping
value: enums.Mapping = driver.configure.signaling.nradio.cell.bwp.ueScheduling.
↪sps.sassignment.uplink.tdomain.chmapping.get(cell_name = 'abc', bwParts =␣
↪repcap.BwParts.Default)
```

Selects the type of PUSCH mapping, for UL configured grant, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
mapping: No help available

set(*cell_name: str*, *mapping: Mapping*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:SASSignment:UL:TDOMain:CHMapping
driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.sassignment.uplink.
↪tdomain.chmapping.set(cell_name = 'abc', mapping = enums.Mapping.A, bwParts =␣
↪repcap.BwParts.Default)
```

Selects the type of PUSCH mapping, for UL configured grant, for BWP <bb>.

param cell_name
No help available

param mapping
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.216 Soffset

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:SASSignment:UL:TDOMain:SOFFset
```

class SoffsetCls

Soffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:SASSignment:UL:TDOMain:SOFFset
value: int = driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.
↪sassignment.uplink.tdomain.soffset.get(cell_name = 'abc', bwParts = repcap.
↪BwParts.Default)
```

Configures the slot offset k2 for the PUSCH, for UL configured grant, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
offset: No help available

set(*cell_name: str*, *offset: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:SASSignment:UL:TDOMain:SOFFset
driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.sassignment.uplink.
↪tdomain.soffset.set(cell_name = 'abc', offset = 1, bwParts = repcap.BwParts.
↪Default)
```

Configures the slot offset k2 for the PUSCH, for UL configured grant, for BWP <bb>.

param cell_name
No help available

param offset
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.217 Symbol

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:SASSignment:UL:TDOMain:SYMBol
```

class SymbolCls

Symbol commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Start_Symbol: int: No parameter help available
- Number_Symbol: int: No parameter help available

get(*cell_name: str*, *bwParts=BwParts.Default*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:SASSignment:UL:TDOMain:SYMBol
value: GetStruct = driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.
↪sassignment.uplink.tdomain.symbol.get(cell_name = 'abc', bwParts = repcap.
↪BwParts.Default)
```

Defines the index of the first allocated OFDM symbol and the number of allocated OFDM symbols, for UL configured grant, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *start_symbol: int*, *number_symbol: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:SASSignment:UL:TDOMain:SYMBol
driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.sassignment.uplink.
↪tdomain.symbol.set(cell_name = 'abc', start_symbol = 1, number_symbol = 1,␣
↪bwParts = repcap.BwParts.Default)
```

Defines the index of the first allocated OFDM symbol and the number of allocated OFDM symbols, for UL configured grant, for BWP <bb>.

param cell_name
No help available

param start_symbol
No help available

param number_symbol
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.218 Uplink

class UplinkCls

Uplink commands group definition. 12 total commands, 12 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.uplink.clone()
```

Subgroups

6.3.4.10.2.219 Alevel

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:UL:ALEVel
```

class AlevelCls

Alevel commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → Level

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:UL:ALEVel
value: enums.Level = driver.configure.signaling.nradio.cell.bwp.ueScheduling.
↪sps.uplink.alevel.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Configures the aggregation level for UL configured grant, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
alevel: No help available

set(*cell_name: str*, *alevel: Level*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:UL:ALEVel
driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.uplink.alevel.
↪set(cell_name = 'abc', alevel = enums.Level.AL1, bwParts = repcap.BwParts.
↪Default)
```

Configures the aggregation level for UL configured grant, for BWP <bb>.

param cell_name
No help available

param alevel
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.220 All

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:UL:ALL
```

class AllCls

All commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Periodicity: enums.SpsPeriodicity: No parameter help available
- Mcs_Table: enums.McsTableB: 256QAM, 64QAM low SE, 64QAM
- Alevel: enums.Level: Aggregation level
- Search_Space_Id: int: No parameter help available
- Resource_Allocation_Type: enums.ResourceAllocationType: Type 0, type 1, dynamic switch
- Rgb_Size: enums.RgbSize: Config 1 or 2
- No_Harq: int: Signaled 'nrofHARQ-Processes'
- Enable_Tp: bool: Signaled 'transformPrecoder'
- Timer: int: Signaled 'configuredGrantTimer'
- Rep_K: enums.PdcchFormatB: No parameter help available
- Rep_Kr_V: enums.Spreset: No parameter help available
- Position: enums.SpsPosition: Signaled 'dmrs-AdditionalPosition'

class SetStruct

Structure for setting input parameters. Contains optional setting parameters. Fields:

- Cell_Name: str: No parameter help available
- Periodicity: enums.SpsPeriodicity: No parameter help available
- Mcs_Table: enums.McsTableB: Optional setting parameter. 256QAM, 64QAM low SE, 64QAM

- Alevel: enums.Level: Optional setting parameter. Aggregation level
- Search_Space_Id: int: No parameter help available
- Resource_Allocation_Type: enums.ResourceAllocationType: Optional setting parameter. Type 0, type 1, dynamic switch
- Rgb_Size: enums.RgbSize: Optional setting parameter. Config 1 or 2
- No_Harq: int: Optional setting parameter. Signaled 'nrofHARQ-Processes'
- Enable_Tp: bool: Optional setting parameter. Signaled 'transformPrecoder'
- Timer: int: Optional setting parameter. Signaled 'configuredGrantTimer'
- Rep_K: enums.PdcchFormatB: No parameter help available
- Rep_Kr_V: enums.Spreset: No parameter help available
- Position: enums.SpsPosition: Optional setting parameter. Signaled 'dmrs-AdditionalPosition'

get(*cell_name: str*, *bwParts=BwParts.Default*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:UL:ALL
value: GetStruct = driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.
↪uplink.all.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Configures several settings for UL configured grant (combination of the other ...SPS:UL:... commands) , for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*structure: SetStruct*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:UL:ALL
structure = driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.uplink.
↪all.SetStruct()
structure.Cell_Name: str = 'abc'
structure.Periodicity: enums.SpsPeriodicity = enums.SpsPeriodicity.S1
structure.Mcs_Table: enums.McsTableB = enums.McsTableB.L64
structure.Alevel: enums.Level = enums.Level.AL1
structure.Search_Space_Id: int = 1
structure.Resource_Allocation_Type: enums.ResourceAllocationType = enums.
↪ResourceAllocationType.DSWich
structure.Rgb_Size: enums.RgbSize = enums.RgbSize.CON1
structure.No_Harq: int = 1
structure.Enable_Tp: bool = False
structure.Timer: int = 1
structure.Rep_K: enums.PdcchFormatB = enums.PdcchFormatB.N1
structure.Rep_Kr_V: enums.Spreset = enums.Spreset.S1
structure.Position: enums.SpsPosition = enums.SpsPosition.POS0
driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.uplink.all.
↪set(structure, bwParts = repcap.BwParts.Default)
```

Configures several settings for UL configured grant (combination of the other ...SPS:UL:... commands) , for BWP <bb>.

param structure
for set value, see the help for SetStruct structure arguments.

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.221 CgTimer

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UEScheduling:SPS:UL:CGTimer
```

class CgTimerCls

CgTimer commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UEScheduling:SPS:UL:CGTimer
value: int = driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.uplink.
↪cgTimer.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Configures the signaled 'configuredGrantTimer' for UL configured grant, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
timer: No help available

set(*cell_name: str*, *timer: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UEScheduling:SPS:UL:CGTimer
driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.uplink.cgTimer.
↪set(cell_name = 'abc', timer = 1, bwParts = repcap.BwParts.Default)
```

Configures the signaled 'configuredGrantTimer' for UL configured grant, for BWP <bb>.

param cell_name
No help available

param timer
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.222 DmrsPosition

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:UL:DMRSposition
```

class DmrsPositionCls

DmrsPosition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → SpsPosition

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:UL:DMRSposition
value: enums.SpsPosition = driver.configure.signaling.nradio.cell.bwp.
↪ueScheduling.sps.uplink.dmrsPosition.get(cell_name = 'abc', bwParts = repcap.
↪BwParts.Default)
```

Configures the signaled 'dmrs-AdditionalPosition' for UL configured grant, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
position: No help available

set(*cell_name: str*, *position: SpsPosition*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:UL:DMRSposition
driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.uplink.dmrsPosition.
↪set(cell_name = 'abc', position = enums.SpsPosition.POS0, bwParts = repcap.
↪BwParts.Default)
```

Configures the signaled 'dmrs-AdditionalPosition' for UL configured grant, for BWP <bb>.

param cell_name
No help available

param position
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.223 McsTable

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:UL:MCSTable
```

class McsTableCls

McsTable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → McsTableB

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:UL:MCSTable
value: enums.McsTableB = driver.configure.signaling.nradio.cell.bwp.
↪ueScheduling.sps.uplink.mcsTable.get(cell_name = 'abc', bwParts = repcap.
↪BwParts.Default)
```

Configures the signaled mcs-Table for UL configured grant, for the initial BWP.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
mcs_table: 256QAM, 64QAM low SE, 64QAM

set(*cell_name: str*, *mcs_table: McsTableB*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:UL:MCSTable
driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.uplink.mcsTable.
↪set(cell_name = 'abc', mcs_table = enums.McsTableB.L64, bwParts = repcap.
↪BwParts.Default)
```

Configures the signaled mcs-Table for UL configured grant, for the initial BWP.

param cell_name
No help available

param mcs_table
256QAM, 64QAM low SE, 64QAM

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.224 Nohp

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:UL:NOHP
```

class NohpCls

Nohp commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:UL:NOHP
value: int = driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.uplink.
↪nohp.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Configures the signaled 'nrofHARQ-Processes' for UL configured grant, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
no_harq: No help available

set(*cell_name: str*, *no_harq: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:UL:NOHP
driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.uplink.nohp.
↪set(cell_name = 'abc', no_harq = 1, bwParts = repcap.BwParts.Default)
```

Configures the signaled 'nrofHARQ-Processes' for UL configured grant, for BWP <bb>.

param cell_name
No help available

param no_harq
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.225 Periodicity

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:UL:PERiodicity
```

class PeriodicityCls

Periodicity commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → SpsPeriodicity

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:UL:PERiodicity
value: enums.SpsPeriodicity = driver.configure.signaling.nradio.cell.bwp.
↪ueScheduling.sps.uplink.periodicity.get(cell_name = 'abc', bwParts = repcap.
↪BwParts.Default)
```

Configures the signaled 'periodicity' for UL configured grant, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
periodicity: SYMn: n symbols Sn: n slots S1K, S1K2, S2K, S5K: 1024, 1280, 2560, 5120 slots

set(*cell_name: str*, *periodicity: SpsPeriodicity*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:UL:PERiodicity
driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.uplink.periodicity.
↪set(cell_name = 'abc', periodicity = enums.SpsPeriodicity.S1, bwParts =␣
↪repcap.BwParts.Default)
```

Configures the signaled 'periodicity' for UL configured grant, for BWP <bb>.

param cell_name
No help available

param periodicity
SYMn: n symbols Sn: n slots S1K, S1K2, S2K, S5K: 1024, 1280, 2560, 5120 slots

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.226 RaType

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:UL:RATYpe
```

class RaTypeCls

RaType commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → ResourceAllocationType

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:UL:RATYpe
value: enums.ResourceAllocationType = driver.configure.signaling.nradio.cell.
↪bwp.ueScheduling.sps.uplink.raType.get(cell_name = 'abc', bwParts = repcap.
↪BwParts.Default)
```

Configures the signaled 'resourceAllocation' for UL configured grant, for BWP <bb>.

param cell_name
type 0, type 1, dynamic switch

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
resource_allocation_type: No help available

`set`(*cell_name: str*, *resource_allocation_type: ResourceAllocationType*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:UL:RATYpe
driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.uplink.raType.
↪set(cell_name = 'abc', resource_allocation_type = enums.
↪ResourceAllocationType.DSWich, bwParts = repcap.BwParts.Default)
```

Configures the signaled 'resourceAllocation' for UL configured grant, for BWP <bb>.

param cell_name
No help available

param resource_allocation_type
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.227 RbgSize

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:UL:RBGSize
```

`class RbgSizeCls`

RbgSize commands group definition. 1 total commands, 0 Subgroups, 1 group commands

`get`(*cell_name: str*, *bwParts=BwParts.Default*) → RgbSize

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:UL:RBGSize
value: enums.RgbSize = driver.configure.signaling.nradio.cell.bwp.ueScheduling.
↪sps.uplink.rbgSize.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Configures the signaled 'rbg-Size' for UL configured grant, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
rgb_size: Config 1 or 2

set(*cell_name: str*, *rgb_size: RgbSize*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:UL:RBGSize
driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.uplink.rbgSize.
↪set(cell_name = 'abc', rgb_size = enums.RgbSize.CON1, bwParts = repcap.
↪BwParts.Default)
```

Configures the signaled 'rbg-Size' for UL configured grant, for BWP <bb>.

param cell_name
No help available

param rgb_size
Config 1 or 2

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.228 RrVersion

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:UL:RRVersion
```

class RrVersionCls

RrVersion commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Rep_K: enums.PdcchFormatB: No parameter help available
- Rep_Kr_V: enums.Spreset: No parameter help available

get(*cell_name: str*, *bwParts=BwParts.Default*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:UL:RRVersion
value: GetStruct = driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.
↪uplink.rrVersion.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Configures the signaled 'repK' and 'repK-RV' for UL configured grant, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *rep_k: PdcchFormatB*, *rep_kr_v: Spreset*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:UL:RRVersion
driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.uplink.rrVersion.
↪set(cell_name = 'abc', rep_k = enums.PdcchFormatB.N1, rep_kr_v = enums.
↪Spreset.S1, bwParts = repcap.BwParts.Default)
```

Configures the signaled 'repK' and 'repK-RV' for UL configured grant, for BWP <bb>.

param cell_name
No help available

param rep_k
No help available

param rep_kr_v
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.229 Ssid

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:UL:SSID
```

class SsidCls

Ssid commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:UL:SSID
value: int = driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.uplink.
↪ssid.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Configures the search space ID for UL configured grant, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
search_space_id: No help available

set(*cell_name: str*, *search_space_id: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:UL:SSID
driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.uplink.ssid.
↪set(cell_name = 'abc', search_space_id = 1, bwParts = repcap.BwParts.Default)
```

Configures the search space ID for UL configured grant, for BWP <bb>.

param cell_name
No help available

param search_space_id
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.230 TpEnable

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:SPS:UL:TPENable
```

class TpEnableCls

TpEnable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:UL:TPENable
value: bool = driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.
↪uplink.tpEnable.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Configures the signaled 'transformPrecoder' for UL configured grant, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
enable_tp: ON: DFT-s-OFDM (with transform precoding) . OFF: CP-OFDM (no transform precoding) .

set(*cell_name: str*, *enable_tp: bool*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:SPS:UL:TPENable
driver.configure.signaling.nradio.cell.bwp.ueScheduling.sps.uplink.tpEnable.
↪set(cell_name = 'abc', enable_tp = False, bwParts = repcap.BwParts.Default)
```

Configures the signaled 'transformPrecoder' for UL configured grant, for BWP <bb>.

param cell_name
No help available

param enable_tp
ON: DFT-s-OFDM (with transform precoding) . OFF: CP-OFDM (no transform precoding) .

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.231 UserDefined

class UserDefinedCls

UserDefined commands group definition. 36 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.clone()
```

Subgroups

6.3.4.10.2.232 CsScheduling

class CsSchedulingCls

CsScheduling commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.
↪csScheduling.clone()
```

Subgroups

6.3.4.10.2.233 K0

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:UDEFined:CSSCheduling:K0
```

class K0Cls

K0 commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:CSSCheduling:K0
value: int or bool = driver.configure.signaling.nradio.cell.bwp.ueScheduling.
↪userDefined.csScheduling.k0.get(cell_name = 'abc', bwParts = repcap.BwParts.
↪Default)
```

Sends a slot offset as ‘minimumSchedulingOffsetK0-r16’ to the UE, for BWP <bb>. The slot offset defines the minimum allowed k0 value (offset between PDCCH and PDSCH, cross-slot scheduling) .

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
k_0: (integer or boolean) integer: Send this value. OFF: Do not send a value.

set(*cell_name: str*, *k_0: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:CSSCheduling:K0
driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.
↪csScheduling.k0.set(cell_name = 'abc', k_0 = 1, bwParts = repcap.BwParts.
↪Default)
```

Sends a slot offset as 'minimumSchedulingOffsetK0-r16' to the UE, for BWP <bb>. The slot offset defines the minimum allowed k0 value (offset between PDCCH and PDSCH, cross-slot scheduling) .

param cell_name
No help available

param k_0
(integer or boolean) integer: Send this value. OFF: Do not send a value.

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.234 K2

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:UDEFined:CSSCheduling:K2
```

class K2Cls

K2 commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:CSSCheduling:K2
value: int or bool = driver.configure.signaling.nradio.cell.bwp.ueScheduling.
↪userDefined.csScheduling.k2.get(cell_name = 'abc', bwParts = repcap.BwParts.
↪Default)
```

Sends a slot offset as 'minimumSchedulingOffsetK2-r16' to the UE, for BWP <bb>. The slot offset defines the minimum allowed k2 value (offset between PDCCH and PUSCH, cross-slot scheduling) .

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
k_2: (integer or boolean) integer: Send this value. OFF: Do not send a value.

set(*cell_name: str*, *k_2: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:CSSCheduling:K2
driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.
↪csScheduling.k2.set(cell_name = 'abc', k_2 = 1, bwParts = repcap.BwParts.
↪Default)
```

Sends a slot offset as 'minimumSchedulingOffsetK2-r16' to the UE, for BWP <bb>. The slot offset defines the minimum allowed k2 value (offset between PDCCH and PUSCH, cross-slot scheduling) .

param cell_name
No help available

param k_2
(integer or boolean) integer: Send this value. OFF: Do not send a value.

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.235 Downlink

class DownlinkCls

Downlink commands group definition. 6 total commands, 6 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.downlink.
↪clone()
```

Subgroups

6.3.4.10.2.236 Alevel

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:UDEFined:DL:ALEVel
```

class AlevelCls

Alevel commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → Level

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:DL:ALEVel
value: enums.Level = driver.configure.signaling.nradio.cell.bwp.ueScheduling.
↪userDefined.downlink.alevel.get(cell_name = 'abc', bwParts = repcap.BwParts.
↪Default)
```

Specifies the aggregation level for the DL, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
level: No help available

set(*cell_name: str*, *level: Level*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:DL:ALEVel
driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.downlink.
↪alevel.set(cell_name = 'abc', level = enums.Level.AL1, bwParts = repcap.
↪BwParts.Default)
```

Specifies the aggregation level for the DL, for BWP <bb>.

param cell_name
No help available

param level
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.237 Bpid

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:UDEFined:DL:BPID
```

class BpidCls

Bpid commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:DL:BPID
value: int = driver.configure.signaling.nradio.cell.bwp.ueScheduling.
↪userDefined.downlink.bpid.get(cell_name = 'abc', bwParts = repcap.BwParts.
↪Default)
```

Queries the ID of the DL bandwidth part that contains the scheduled allocation, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
idn: No help available

6.3.4.10.2.238 McsTable

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:UDEFined:DL:MCSTable
```

class McsTableCls

McsTable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → McsTableB

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:DL:MCSTable
value: enums.McsTableB = driver.configure.signaling.nradio.cell.bwp.
↪ueScheduling.userDefined.downlink.mcsTable.get(cell_name = 'abc', bwParts =␣
↪repcap.BwParts.Default)
```

Selects the MCS table that the UE must use to determine the modulation order and the target code rate of the PDSCH, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
mcs_table: 256QAM, 64QAM low SE, 64QAM

set(*cell_name: str*, *mcs_table: McsTableB*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:DL:MCSTable
driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.downlink.
↪mcsTable.set(cell_name = 'abc', mcs_table = enums.McsTableB.L64, bwParts =␣
↪repcap.BwParts.Default)
```

Selects the MCS table that the UE must use to determine the modulation order and the target code rate of the PDSCH, for BWP <bb>.

param cell_name
No help available

param mcs_table
256QAM, 64QAM low SE, 64QAM

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.239 Padding

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:UDEFined:DL:PADDing
```

class PaddingCls

Padding commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:DL:PADDing
value: bool = driver.configure.signaling.nradio.cell.bwp.ueScheduling.
↪userDefined.downlink.padding.get(cell_name = 'abc', bwParts = repcap.BwParts.
↪Default)
```

Activates or deactivates downlink padding at the MAC layer, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
enable: No help available

set(*cell_name: str*, *enable: bool*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:DL:PADDing
driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.downlink.
↪padding.set(cell_name = 'abc', enable = False, bwParts = repcap.BwParts.
↪Default)
```

Activates or deactivates downlink padding at the MAC layer, for BWP <bb>.

param cell_name
No help available

param enable
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.240 Ssid

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:UDEFined:DL:SSID
```

class SsidCls

Ssid commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:DL:SSID
value: int = driver.configure.signaling.nradio.cell.bwp.ueScheduling.
↪userDefined.downlink.ssid.get(cell_name = 'abc', bwParts = repcap.BwParts.
↪Default)
```

Queries the ID of the search space for the DL, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
idn: No help available

6.3.4.10.2.241 VpMapping

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:UDEFined:DL:VPMapping
```

class VpMappingCls

VpMapping commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → MappingI

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:DL:VPMapping
value: enums.MappingI = driver.configure.signaling.nradio.cell.bwp.ueScheduling.
↪userDefined.downlink.vpMapping.get(cell_name = 'abc', bwParts = repcap.
↪BwParts.Default)
```

Selects whether interleaved or non-interleaved virtual RB to physical RB mapping is applied to the PDSCH, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
mapping: Interleaved or non-interleaved

set(*cell_name: str*, *mapping: MappingI*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:DL:VPMapping
driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.downlink.
↪vpMapping.set(cell_name = 'abc', mapping = enums.MappingI.INT, bwParts =␣
↪repcap.BwParts.Default)
```

Selects whether interleaved or non-interleaved virtual RB to physical RB mapping is applied to the PDSCH, for BWP <bb>.

param cell_name
No help available

param mapping
Interleaved or non-interleaved

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.242 Sassignment

class SassignmentCls

Sassignment commands group definition. 21 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.sassignment.
↪clone()
```

Subgroups

6.3.4.10.2.243 Downlink

class DownlinkCls

Downlink commands group definition. 10 total commands, 7 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.sassignment.
↪downlink.clone()
```

Subgroups

6.3.4.10.2.244 All

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:UDEFined:SASSignment:DL:ALL
```

class AllCls

All commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Slot: List[int]: Index number of the slot
- Enable: List[bool]: No parameter help available
- Number_Rb: List[int]: No parameter help available
- Start_Rb: List[int]: No parameter help available
- Mcs: List[int]: No parameter help available
- Dci_Format: List[enums.DciFormatB]: No parameter help available
- Mimo: List[enums.Mimo]: No parameter help available

class SetStruct

Structure for setting input parameters. Fields:

- Cell_Name: str: No parameter help available
- Slot: List[int]: Index number of the slot
- Enable: List[bool]: No parameter help available
- Number_Rb: List[int]: No parameter help available
- Start_Rb: List[int]: No parameter help available
- Mcs: List[int]: No parameter help available
- Dci_Format: List[enums.DciFormatB]: No parameter help available
- Mimo: List[enums.Mimo]: No parameter help available

get(*cell_name: str*, *bwParts=BwParts.Default*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:DL:ALL
value: GetStruct = driver.configure.signaling.nradio.cell.bwp.ueScheduling.
↪userDefined.sassignment.downlink.all.get(cell_name = 'abc', bwParts = repcap.
↪BwParts.Default)
```

Defines scheduling settings for one or more DL slots, for BWP <bb>. The parameter sequence contains one set of values per slot: <CellName>, {<Slot>, <Enable>, <NumberRB>, <StartRB>, <MCS>, <DCI-Format>, <Mimo>}slot a, {...}slot b, ... A query returns all DL slots.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*structure: SetStruct*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:DL:ALL
structure = driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.
↪sassignment.downlink.all.SetStruct()
structure.Cell_Name: str = 'abc'
structure.Slot: List[int] = [1, 2, 3]
structure.Enable: List[bool] = [True, False, True]
structure.Number_Rb: List[int] = [1, 2, 3]
structure.Start_Rb: List[int] = [1, 2, 3]
structure.Mcs: List[int] = [1, 2, 3]
structure.Dci_Format: List[enums.DciFormatB] = [DciFormatB.D10, DciFormatB.D11]
structure.Mimo: List[enums.Mimo] = [Mimo.M22, Mimo.SISO]
driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.sassignment.
↪downlink.all.set(structure, bwParts = repcap.BwParts.Default)
```

Defines scheduling settings for one or more DL slots, for BWP <bb>. The parameter sequence contains one set of values per slot: <CellName>, {<Slot>, <Enable>, <NumberRB>, <StartRB>, <MCS>, <DCI-Format>, <Mimo>}slot a, {...}slot b, ... A query returns all DL slots.

param structure
for set value, see the help for SetStruct structure arguments.

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.245 DciFormat

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:DL:DCIFormat
```

class DciFormatCls

DciFormat commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *slot: int*, *bwParts=BwParts.Default*) → DciFormatB

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:DL:DCIFormat
value: enums.DciFormatB = driver.configure.signaling.nradio.cell.bwp.
↪ueScheduling.userDefined.sassignment.downlink.dciFormat.get(cell_name = 'abc',
↪ slot = 1, bwParts = repcap.BwParts.Default)
```

Defines the DCI format for the DL slot with the index <Slot>, for BWP <bb>.

param cell_name
No help available

param slot
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
dci_format: No help available

set(*cell_name: str*, *slot: int*, *dci_format: DciFormatB*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:DL:DCIFormat
driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.sassignment.
↪downlink.dciFormat.set(cell_name = 'abc', slot = 1, dci_format = enums.
↪DciFormatB.D10, bwParts = repcap.BwParts.Default)
```

Defines the DCI format for the DL slot with the index <Slot>, for BWP <bb>.

param cell_name
No help available

param slot
No help available

param dci_format
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.246 Enable

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:UDEFined:SASSignment:DL:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *slot: int*, *bwParts=BwParts.Default*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:DL:ENABle
value: bool = driver.configure.signaling.nradio.cell.bwp.ueScheduling.
↪userDefined.sassignment.downlink.enable.get(cell_name = 'abc', slot = 1,␣
↪bwParts = repcap.BwParts.Default)
```

Enables or disables scheduling of the DL slot with the index <Slot>, for BWP <bb>.

param cell_name
No help available

param slot
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
enable: No help available

set(*cell_name: str*, *slot: int*, *enable: bool*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:DL:ENABle
driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.sassignment.
↪downlink.enable.set(cell_name = 'abc', slot = 1, enable = False, bwParts =␣
↪repcap.BwParts.Default)
```

Enables or disables scheduling of the DL slot with the index <Slot>, for BWP <bb>.

param cell_name
No help available

param slot
No help available

param enable
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.247 Mcs

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:UDEFined:SASSignment:DL:MCS
```

class McsCls

Mcs commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *slot: int*, *bwParts=BwParts.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:DL:MCS
value: int = driver.configure.signaling.nradio.cell.bwp.ueScheduling.
↪userDefined.sassignment.downlink.mcs.get(cell_name = 'abc', slot = 1, bwParts␣
↪= repcap.BwParts.Default)
```

Specifies the MCS index for the DL slot with the index <Slot>, for BWP <bb>.

param cell_name
No help available

param slot
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
mcs: No help available

set(*cell_name: str*, *slot: int*, *mcs: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:DL:MCS
driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.sassignment.
↪downlink.mcs.set(cell_name = 'abc', slot = 1, mcs = 1, bwParts = repcap.
↪BwParts.Default)
```

Specifies the MCS index for the DL slot with the index <Slot>, for BWP <bb>.

param cell_name
No help available

param slot
No help available

param mcs
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.248 Mimo

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:UDEFined:SASSignment:DL:MIMO
```

class MimoCls

Mimo commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *slot: int*, *bwParts=BwParts.Default*) → Mimo

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:DL:MIMO
value: enums.Mimo = driver.configure.signaling.nradio.cell.bwp.ueScheduling.
↪userDefined.sassignment.downlink.mimo.get(cell_name = 'abc', slot = 1,␣
↪bwParts = repcap.BwParts.Default)
```

Defines the DCI format for the DL slot with the index <Slot>, for BWP <bb>.

param cell_name
No help available

param slot
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
mimo: SISO: 1xN M22: 2xN M33: 3xN M44: 4xN

set(*cell_name: str*, *slot: int*, *mimo: Mimo*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:DL:MIMO
driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.sassignment.
↪downlink.mimo.set(cell_name = 'abc', slot = 1, mimo = enums.Mimo.M22, bwParts␣
↪= repcap.BwParts.Default)
```

Defines the DCI format for the DL slot with the index <Slot>, for BWP <bb>.

param cell_name
No help available

param slot
No help available

param mimo
SISO: 1xN M22: 2xN M33: 3xN M44: 4xN

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.249 Rb

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:UDEFined:SASSignment:DL:RB
```

class RbCls

Rb commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Number_Rb: int: No parameter help available

- Start_Rb: int: No parameter help available

get(*cell_name: str*, *slot: int*, *bwParts=BwParts.Default*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:DL:RB
value: GetStruct = driver.configure.signaling.nradio.cell.bwp.ueScheduling.
↪userDefined.sassignment.downlink.rb.get(cell_name = 'abc', slot = 1, bwParts␣
↪= repcap.BwParts.Default)
```

Specifies the scheduled RB allocation for the DL slot with the index <Slot>, for BWP <bb>.

param cell_name
No help available

param slot
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *slot: int*, *number_rb: int*, *start_rb: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:DL:RB
driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.sassignment.
↪downlink.rb.set(cell_name = 'abc', slot = 1, number_rb = 1, start_rb = 1,␣
↪bwParts = repcap.BwParts.Default)
```

Specifies the scheduled RB allocation for the DL slot with the index <Slot>, for BWP <bb>.

param cell_name
No help available

param slot
No help available

param number_rb
No help available

param start_rb
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.250 Tdomain

class TdomainCls

Tdomain commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.sassignment.
↪downlink.tdomain.clone()
```

Subgroups

6.3.4.10.2.251 AnsOffset

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:DL:TDOMain:ANSoffset
```

class AnsOffsetCls

AnsOffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *slot: int*, *bwParts=BwParts.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:DL:TDOMain:ANSoffset
value: int = driver.configure.signaling.nradio.cell.bwp.ueScheduling.
↪userDefined.sassignment.downlink.tdomain.ansOffset.get(cell_name = 'abc',␣
↪slot = 1, bwParts = repcap.BwParts.Default)
```

Configures the ACK/NACK slot offset k1, for the DL slot with the index <Slot>, for BWP <bb>.

param cell_name
No help available

param slot
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
offset: No help available

set(*cell_name: str*, *slot: int*, *offset: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:DL:TDOMain:ANSoffset
driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.sassignment.
↪downlink.tdomain.ansOffset.set(cell_name = 'abc', slot = 1, offset = 1,␣
↪bwParts = repcap.BwParts.Default)
```

Configures the ACK/NACK slot offset k1, for the DL slot with the index <Slot>, for BWP <bb>.

param cell_name
No help available

param slot
No help available

param offset
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.252 Chmapping

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:DL:TDOMain:CHMapping
```

class ChmappingCls

Chmapping commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *slot: int*, *bwParts=BwParts.Default*) → Mapping

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:DL:TDOMain:CHMapping
value: enums.Mapping = driver.configure.signaling.nradio.cell.bwp.ueScheduling.
↪userDefined.sassignment.downlink.tdomain.chmapping.get(cell_name = 'abc',␣
↪slot = 1, bwParts = repcap.BwParts.Default)
```

Selects the type of PDSCH mapping, for the DL slot with the index <Slot>, for BWP <bb>.

param cell_name
No help available

param slot
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
mapping: No help available

set(*cell_name: str*, *slot: int*, *mapping: Mapping*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:DL:TDOMain:CHMapping
driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.sassignment.
↪downlink.tdomain.chmapping.set(cell_name = 'abc', slot = 1, mapping = enums.
↪Mapping.A, bwParts = repcap.BwParts.Default)
```

Selects the type of PDSCH mapping, for the DL slot with the index <Slot>, for BWP <bb>.

param cell_name
No help available

param slot
No help available

param mapping
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.253 Soffset

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:DL:TDOMain:SOFFset
```

class SoffsetCls

Soffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *slot: int*, *bwParts=BwParts.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:DL:TDOMain:SOFFset
value: int = driver.configure.signaling.nradio.cell.bwp.ueScheduling.
↪userDefined.sassignment.downlink.tdomain.soffset.get(cell_name = 'abc', slot␣
↪= 1, bwParts = repcap.BwParts.Default)
```

Configures the slot offset k0 for the PDSCH, for the DL slot with the index <Slot>, for BWP <bb>.

param cell_name
No help available

param slot
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
offset: No help available

set(*cell_name: str*, *slot: int*, *offset: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:DL:TDOMain:SOFFset
driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.sassignment.
↪downlink.tdomain.soffset.set(cell_name = 'abc', slot = 1, offset = 1, bwParts␣
↪= repcap.BwParts.Default)
```

Configures the slot offset k0 for the PDSCH, for the DL slot with the index <Slot>, for BWP <bb>.

param cell_name
No help available

param slot
No help available

param offset
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.254 Symbol

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:DL:TDOMain:SYMBol
```

class SymbolCls

Symbol commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Number_Symbol: int: No parameter help available
- Start_Symbol: int: No parameter help available

get(*cell_name: str*, *slot: int*, *bwParts=BwParts.Default*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:DL:TDOMain:SYMBol
value: GetStruct = driver.configure.signaling.nradio.cell.bwp.ueScheduling.
↪userDefined.sassignment.downlink.tdomain.symbol.get(cell_name = 'abc', slot =␣
↪1, bwParts = repcap.BwParts.Default)
```

Defines the number of allocated OFDM symbols and the index of the first allocated OFDM symbol for the PDSCH, for the DL slot with the index <Slot>, for BWP <bb>.

param cell_name
No help available

param slot
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *slot: int*, *number_symbol: int*, *start_symbol: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:DL:TDOMain:SYMBol
driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.sassignment.
```

(continues on next page)

(continued from previous page)

```
↪downlink.tdomain.symbol.set(cell_name = 'abc', slot = 1, number_symbol = 1,␣
↪start_symbol = 1, bwParts = repcap.BwParts.Default)
```

Defines the number of allocated OFDM symbols and the index of the first allocated OFDM symbol for the PDSCH, for the DL slot with the index <Slot>, for BWP <bb>.

param cell_name
No help available

param slot
No help available

param number_symbol
No help available

param start_symbol
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.255 Uplink

class UplinkCls

Uplink commands group definition. 11 total commands, 8 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.sassignment.
↪uplink.clone()
```

Subgroups

6.3.4.10.2.256 All

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:UDEFined:SASSignment:UL:ALL
```

class AllCls

All commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Slot: List[int]: Index number of the slot
- Enable: List[bool]: No parameter help available
- Number_Rb: List[int]: No parameter help available

- Start_Rb: List[int]: No parameter help available
- Mcs: List[int]: No parameter help available
- Dci_Format: List[enums.DciFormatC]: No parameter help available
- Mimo: List[enums.MimoB]: No parameter help available

class SetStruct

Structure for setting input parameters. Fields:

- Cell_Name: str: No parameter help available
- Slot: List[int]: Index number of the slot
- Enable: List[bool]: No parameter help available
- Number_Rb: List[int]: No parameter help available
- Start_Rb: List[int]: No parameter help available
- Mcs: List[int]: No parameter help available
- Dci_Format: List[enums.DciFormatC]: No parameter help available
- Mimo: List[enums.MimoB]: No parameter help available

get(*cell_name: str*, *bwParts=BwParts.Default*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:UL:ALL
value: GetStruct = driver.configure.signaling.nradio.cell.bwp.ueScheduling.
↪userDefined.sassignment.uplink.all.get(cell_name = 'abc', bwParts = repcap.
↪BwParts.Default)
```

Defines scheduling settings for one or more UL slots, for BWP <bb>. The parameter sequence contains one set of values per slot: <CellName>, {<Slot>, <Enable>, <NumberRB>, <StartRB>, <MCS>, <DCI-Format>, <Mimo>}slot a, {...}slot b, ... A query returns all UL slots.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*structure: SetStruct*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:UL:ALL
structure = driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.
↪sassignment.uplink.all.SetStruct()
structure.Cell_Name: str = 'abc'
structure.Slot: List[int] = [1, 2, 3]
structure.Enable: List[bool] = [True, False, True]
structure.Number_Rb: List[int] = [1, 2, 3]
structure.Start_Rb: List[int] = [1, 2, 3]
structure.Mcs: List[int] = [1, 2, 3]
```

(continues on next page)

(continued from previous page)

```
structure.Dci_Format: List[enums.DciFormatC] = [DciFormatC.D00, DciFormatC.D01]
structure.Mimo: List[enums.MimoB] = [MimoB.M22, MimoB.SISO]
driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.sassignment.
↪uplink.all.set(structure, bwParts = repcap.BwParts.Default)
```

Defines scheduling settings for one or more UL slots, for BWP <bb>. The parameter sequence contains one set of values per slot: <CellName>, {<Slot>, <Enable>, <NumberRB>, <StartRB>, <MCS>, <DCI-Format>, <Mimo>}slot a, {...}slot b, ... A query returns all UL slots.

param structure
for set value, see the help for SetStruct structure arguments.

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.257 DciFormat

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:UL:DCIFormat
```

class DciFormatCls

DciFormat commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *slot: int*, *bwParts=BwParts.Default*) → DciFormatC

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:UL:DCIFormat
value: enums.DciFormatC = driver.configure.signaling.nradio.cell.bwp.
↪ueScheduling.userDefined.sassignment.uplink.dciFormat.get(cell_name = 'abc',␣
↪slot = 1, bwParts = repcap.BwParts.Default)
```

Defines the DCI format for the UL slot with the index <Slot>, for BWP <bb>.

param cell_name
No help available

param slot
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
dci_format: No help available

set(*cell_name: str*, *slot: int*, *dci_format: DciFormatC*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:UL:DCIFormat
driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.sassignment.
```

(continues on next page)

(continued from previous page)

```
↪uplink.dciFormat.set(cell_name = 'abc', slot = 1, dci_format = enums.
↪DciFormatC.D00, bwParts = repcap.BwParts.Default)
```

Defines the DCI format for the UL slot with the index <Slot>, for BWP <bb>.

param cell_name
No help available

param slot
No help available

param dci_format
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.258 Enable

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:UDEFined:SASSignment:UL:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *slot: int*, *bwParts=BwParts.Default*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:UL:ENABle
value: bool = driver.configure.signaling.nradio.cell.bwp.ueScheduling.
↪userDefined.sassignment.uplink.enable.get(cell_name = 'abc', slot = 1,␣
↪bwParts = repcap.BwParts.Default)
```

Enables or disables scheduling of the UL slot with the index <Slot>, for BWP <bb>.

param cell_name
No help available

param slot
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
enable: No help available

set(*cell_name: str*, *slot: int*, *enable: bool*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:UL:ENABle
driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.sassignment.
```

(continues on next page)

(continued from previous page)

```
↪uplink.enable.set(cell_name = 'abc', slot = 1, enable = False, bwParts =␣
↪repcap.BwParts.Default)
```

Enables or disables scheduling of the UL slot with the index <Slot>, for BWP <bb>.

param cell_name
No help available

param slot
No help available

param enable
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.259 Mcs

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:UDEFined:SASSignment:UL:MCS
```

class McsCls

Mcs commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *slot: int*, *bwParts=BwParts.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:UL:MCS
value: int = driver.configure.signaling.nradio.cell.bwp.ueScheduling.
↪userDefined.sassignment.uplink.mcs.get(cell_name = 'abc', slot = 1, bwParts =␣
↪repcap.BwParts.Default)
```

Specifies the MCS index for the UL slot with the index <Slot>, for BWP <bb>.

param cell_name
No help available

param slot
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
mcs: No help available

set(*cell_name: str*, *slot: int*, *mcs: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:UL:MCS
driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.sassignment.
```

(continues on next page)

(continued from previous page)

```
↪uplink.mcs.set(cell_name = 'abc', slot = 1, mcs = 1, bwParts = repcap.BwParts.
↪Default)
```

Specifies the MCS index for the UL slot with the index <Slot>, for BWP <bb>.

param cell_name
No help available

param slot
No help available

param mcs
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.260 Mimo

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:UDEFined:SASSignment:UL:MIMO
```

class MimoCls

Mimo commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *slot: int*, *bwParts=BwParts.Default*) → MimoB

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:UL:MIMO
value: enums.MimoB = driver.configure.signaling.nradio.cell.bwp.ueScheduling.
↪userDefined.sassignment.uplink.mimo.get(cell_name = 'abc', slot = 1, bwParts␣
↪= repcap.BwParts.Default)
```

Specifies the MIMO scheme for the UL slot with the index <Slot>, for BWP <bb>.

param cell_name
No help available

param slot
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
mimo: SISO: nx1 M22: nx2

set(*cell_name: str*, *slot: int*, *mimo: MimoB*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:UL:MIMO
driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.sassignment.
```

(continues on next page)

(continued from previous page)

```
↪uplink.mimo.set(cell_name = 'abc', slot = 1, mimo = enums.MimoB.M22, bwParts␣
↪= repcap.BwParts.Default)
```

Specifies the MIMO scheme for the UL slot with the index <Slot>, for BWP <bb>.

param cell_name
No help available

param slot
No help available

param mimo
SISO: nx1 M22: nx2

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.261 Rb

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:UDEFined:SASSignment:UL:RB
```

class RbCls

Rb commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Number_Rb: int: No parameter help available
- Start_Rb: int: No parameter help available

get(*cell_name: str*, *slot: int*, *bwParts=BwParts.Default*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:UL:RB
value: GetStruct = driver.configure.signaling.nradio.cell.bwp.ueScheduling.
↪userDefined.sassignment.uplink.rb.get(cell_name = 'abc', slot = 1, bwParts =␣
↪repcap.BwParts.Default)
```

Specifies the scheduled RB allocation for the UL slot with the index <Slot>, for BWP <bb>.

param cell_name
No help available

param slot
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *slot: int*, *number_rb: int*, *start_rb: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:UL:RB
driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.sassignment.
↪uplink.rb.set(cell_name = 'abc', slot = 1, number_rb = 1, start_rb = 1,␣
↪bwParts = repcap.BwParts.Default)
```

Specifies the scheduled RB allocation for the UL slot with the index <Slot>, for BWP <bb>.

param cell_name
No help available

param slot
No help available

param number_rb
No help available

param start_rb
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.262 Tdomain

class TdomainCls

Tdomain commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.sassignment.
↪uplink.tdomain.clone()
```

Subgroups

6.3.4.10.2.263 Chmapping

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:UL:TDOMain:CHMapping
```

class ChmappingCls

Chmapping commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *slot: int*, *bwParts=BwParts.Default*) → Mapping

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:UL:TDOMain:CHMapping
value: enums.Mapping = driver.configure.signaling.nradio.cell.bwp.ueScheduling.
↪userDefined.sassignment.uplink.tdomain.chmapping.get(cell_name = 'abc', slot␣
↪= 1, bwParts = repcap.BwParts.Default)
```

Selects the type of PUSCH mapping, for the UL slot with the index <Slot>, for BWP <bb>.

param cell_name
No help available

param slot
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
mapping: No help available

set(*cell_name: str*, *slot: int*, *mapping: Mapping*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:UL:TDOMain:CHMapping
driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.sassignment.
↪uplink.tdomain.chmapping.set(cell_name = 'abc', slot = 1, mapping = enums.
↪Mapping.A, bwParts = repcap.BwParts.Default)
```

Selects the type of PUSCH mapping, for the UL slot with the index <Slot>, for BWP <bb>.

param cell_name
No help available

param slot
No help available

param mapping
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.264 PnoRepet

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:UL:TDOMain:PNORepet
```

class PnoRepetCls

PnoRepet commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *slot: int*, *bwParts=BwParts.Default*) → Repetitions

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:UL:TDOMain:PNORepet
value: enums.Repetitions = driver.configure.signaling.nradio.cell.bwp.
↪ueScheduling.userDefined.sassignment.uplink.tdomain.pnoRepet.get(cell_name =
↪'abc', slot = 1, bwParts = repcap.BwParts.Default)
```

Specifies the number of PUSCH repetitions signaled as 'numberOfRepetitions', for the UL slot with the index <Slot>, for BWP <bb>. Prerequisite: Configure [CONFigure:]SIGNaling:NRADio:CELL:BWP<bb>:UESCheduling:UDEFined:UL:PRTYpe.

param cell_name
No help available

param slot
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
repetitions: No help available

set(*cell_name: str*, *slot: int*, *repetitions: Repetitions*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:UL:TDOMain:PNORepet
driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.sassignment.
↪uplink.tdomain.pnoRepet.set(cell_name = 'abc', slot = 1, repetitions = enums.
↪Repetitions.N12, bwParts = repcap.BwParts.Default)
```

Specifies the number of PUSCH repetitions signaled as 'numberOfRepetitions', for the UL slot with the index <Slot>, for BWP <bb>. Prerequisite: Configure [CONFigure:]SIGNaling:NRADio:CELL:BWP<bb>:UESCheduling:UDEFined:UL:PRTYpe.

param cell_name
No help available

param slot
No help available

param repetitions
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.265 Soffset

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:UL:TDOMain:SOFFset
```

class SoffsetCls

Soffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *slot: int*, *bwParts=BwParts.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:UL:TDOMain:SOFFset
value: int = driver.configure.signaling.nradio.cell.bwp.ueScheduling.
↪userDefined.sassignment.uplink.tdomain.soffset.get(cell_name = 'abc', slot =␣
↪1, bwParts = repcap.BwParts.Default)
```

Configures the slot offset k2 for the PUSCH, for the UL slot with the index <Slot>, for BWP <bb>.

param cell_name
No help available

param slot
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
offset: No help available

set(*cell_name: str*, *slot: int*, *offset: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:UL:TDOMain:SOFFset
driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.sassignment.
↪uplink.tdomain.soffset.set(cell_name = 'abc', slot = 1, offset = 1, bwParts =␣
↪repcap.BwParts.Default)
```

Configures the slot offset k2 for the PUSCH, for the UL slot with the index <Slot>, for BWP <bb>.

param cell_name
No help available

param slot
No help available

param offset
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.266 Symbol

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:UL:TDOMain:SYMBol
```

class SymbolCls

Symbol commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Number_Symbol: int: No parameter help available
- Start_Symbol: int: No parameter help available

get(*cell_name: str*, *slot: int*, *bwParts=BwParts.Default*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:UL:TDOMain:SYMBol
value: GetStruct = driver.configure.signaling.nradio.cell.bwp.ueScheduling.
↪userDefined.sassignment.uplink.tdomain.symbol.get(cell_name = 'abc', slot = 1,
↪ bwParts = repcap.BwParts.Default)
```

Defines the number of allocated OFDM symbols and the index of the first allocated OFDM symbol for the PDSCH, for the UL slot with the index <Slot>, for BWP <bb>.

param cell_name
No help available

param slot
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *slot: int*, *number_symbol: int*, *start_symbol: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:UL:TDOMain:SYMBol
driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.sassignment.
↪uplink.tdomain.symbol.set(cell_name = 'abc', slot = 1, number_symbol = 1,␣
↪start_symbol = 1, bwParts = repcap.BwParts.Default)
```

Defines the number of allocated OFDM symbols and the index of the first allocated OFDM symbol for the PDSCH, for the UL slot with the index <Slot>, for BWP <bb>.

param cell_name
No help available

param slot
No help available

param number_symbol
No help available

param start_symbol
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.267 Tpmi

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:UDEFined:SASSignment:UL:TPMI
```

class TpmiCls

Tpmi commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *slot: int*, *bwParts=BwParts.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:UL:TPMI
value: int = driver.configure.signaling.nradio.cell.bwp.ueScheduling.
↪userDefined.sassignment.uplink.tpmi.get(cell_name = 'abc', slot = 1, bwParts␣
↪= repcap.BwParts.Default)
```

Specifies the TPMI for the UL slot with the index <Slot>, for BWP <bb>.

param cell_name
No help available

param slot
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
tpmi: No help available

set(*cell_name: str*, *slot: int*, *tpmi: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:SASSignment:UL:TPMI
driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.sassignment.
↪uplink.tpmi.set(cell_name = 'abc', slot = 1, tpmi = 1, bwParts = repcap.
↪BwParts.Default)
```

Specifies the TPMI for the UL slot with the index <Slot>, for BWP <bb>.

param cell_name
No help available

param slot
No help available

param tpmi
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.268 Uplink

class UplinkCls

Uplink commands group definition. 7 total commands, 7 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.uplink.
↪clone()
```

Subgroups

6.3.4.10.2.269 Alevel

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:UDEFined:UL:ALEVel
```

class AlevelCls

Alevel commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → Level

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:UL:ALEVel
value: enums.Level = driver.configure.signaling.nradio.cell.bwp.ueScheduling.
↪userDefined.uplink.alevel.get(cell_name = 'abc', bwParts = repcap.BwParts.
↪Default)
```

Specifies the aggregation level for the UL, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
level: No help available

set(*cell_name: str*, *level: Level*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:UL:ALEVel
driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.uplink.
↪alevel.set(cell_name = 'abc', level = enums.Level.AL1, bwParts = repcap.
↪BwParts.Default)
```

Specifies the aggregation level for the UL, for BWP <bb>.

param cell_name
No help available

param level
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.270 Bpid

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:UDEFined:UL:BPID
```

class BpidCls

Bpid commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:UL:BPID
value: int = driver.configure.signaling.nradio.cell.bwp.ueScheduling.
↪userDefined.uplink.bpid.get(cell_name = 'abc', bwParts = repcap.BwParts.
↪Default)
```

Queries the ID of the UL bandwidth part that contains the scheduled allocation, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
idn: No help available

6.3.4.10.2.271 McsTable

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:UDEFined:UL:MCSTable
```

class McsTableCls

McsTable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → McsTableB

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:UL:MCSTable
value: enums.McsTableB = driver.configure.signaling.nradio.cell.bwp.
↪ueScheduling.userDefined.uplink.mcsTable.get(cell_name = 'abc', bwParts =␣
↪repcap.BwParts.Default)
```

Defines which MCS table must be used for PUSCH without transform precoding, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
mcs_table: 256QAM, 64QAM low SE, 64QAM

set(*cell_name: str*, *mcs_table: McsTableB*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:UL:MCSTable
driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.uplink.
↪mcsTable.set(cell_name = 'abc', mcs_table = enums.McsTableB.L64, bwParts =␣
↪repcap.BwParts.Default)
```

Defines which MCS table must be used for PUSCH without transform precoding, for BWP <bb>.

param cell_name
No help available

param mcs_table
256QAM, 64QAM low SE, 64QAM

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.272 PaFactor

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:UDEFined:UL:PAFactor
```

class PaFactorCls

PaFactor commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → UeScFactor

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:UL:PAFactor
value: enums.UeScFactor = driver.configure.signaling.nradio.cell.bwp.
↪ueScheduling.userDefined.uplink.paFactor.get(cell_name = 'abc', bwParts =␣
↪repcap.BwParts.Default)
```

Defines the 'pusch-AggregationFactor' of the PUSCH configuration, signaled to the UE, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
factor: No help available

set(*cell_name: str*, *factor: UeScFactor*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:UL:PAFactor
driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.uplink.
↪paFactor.set(cell_name = 'abc', factor = enums.UeScFactor.N2, bwParts =␣
↪repcap.BwParts.Default)
```

Defines the 'pusch-AggregationFactor' of the PUSCH configuration, signaled to the UE, for BWP <bb>.

param cell_name
No help available

param factor
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.273 PnoRepet

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:UDEFined:UL:PNORepet
```

class PnoRepetCls

PnoRepet commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → Repetitions

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:UL:PNORepet
value: enums.Repetitions = driver.configure.signaling.nradio.cell.bwp.
↪ueScheduling.userDefined.uplink.pnoRepet.get(cell_name = 'abc', bwParts =␣
↪repcap.BwParts.Default)
```

No command help available

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
repetitions: No help available

set(*cell_name: str*, *repetitions: Repetitions*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:UL:PNORepet
driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.uplink.
↪pnoRepet.set(cell_name = 'abc', repetitions = enums.Repetitions.N12, bwParts␣
↪= repcap.BwParts.Default)
```

No command help available

param cell_name
No help available

param repetitions
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.274 Prtype

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:UDEFined:UL:PRTYpe
```

class PrtypeCls

Prtype commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → Prtype

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:UL:PRTYpe
value: enums.Prtype = driver.configure.signaling.nradio.cell.bwp.ueScheduling.
↪userDefined.uplink.prtype.get(cell_name = 'abc', bwParts = repcap.BwParts.
↪Default)
```

Specifies the PUSCH repetition type signaled as 'pusch-RepTypeIndicatorDCI-0-1', for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
type_py: Not signaled, type A, type B.

set(*cell_name: str*, *type_py: Prtype*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:UL:PRTYpe
driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.uplink.
↪prtype.set(cell_name = 'abc', type_py = enums.Prtype.OFF, bwParts = repcap.
↪BwParts.Default)
```

Specifies the PUSCH repetition type signaled as 'pusch-RepTypeIndicatorDCI-0-1', for BWP <bb>.

param cell_name
No help available

param type_py
Not signaled, type A, type B.

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.275 Ssid

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:UDEFined:UL:SSID
```

class SsidCls

Ssid commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:UL:SSID
value: int = driver.configure.signaling.nradio.cell.bwp.ueScheduling.
↪userDefined.uplink.ssid.get(cell_name = 'abc', bwParts = repcap.BwParts.
↪Default)
```

Configures the ID of the search space for the UL, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
idn: No help available

set(*cell_name: str*, *idn: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:UESCheduling:UDEFined:UL:SSID
driver.configure.signaling.nradio.cell.bwp.ueScheduling.userDefined.uplink.ssid.
↪set(cell_name = 'abc', idn = 1, bwParts = repcap.BwParts.Default)
```

Configures the ID of the search space for the UL, for BWP <bb>.

param cell_name
No help available

param idn
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.276 Uplink

class UplinkCls

Uplink commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.bwp.uplink.clone()
```

Subgroups

6.3.4.10.2.277 LbWidth

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwpid>:UL:LBWidth
```

class LbWidthCls

LbWidth commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwpid>:UL:LBWidth
value: int = driver.configure.signaling.nradio.cell.bwp.uplink.lbWidth.get(cell_
↪name = 'abc', bwParts = repcap.BwParts.Default)
```

Defines the resource indication value (RIV) signaled as 'locationAndBandwidth', for the uplink, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
riv: No help available

set(*cell_name: str*, *riv: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwpid>:UL:LBWidth
driver.configure.signaling.nradio.cell.bwp.uplink.lbWidth.set(cell_name = 'abc',
↪ riv = 1, bwParts = repcap.BwParts.Default)
```

Defines the resource indication value (RIV) signaled as 'locationAndBandwidth', for the uplink, for BWP <bb>.

param cell_name
No help available

param riv
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.278 Mode

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwpid>:UL:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → ConfigMode

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwpid>:UL:MODE
value: enums.ConfigMode = driver.configure.signaling.nradio.cell.bwp.uplink.
↪mode.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Selects a configuration mode for the UL BWP settings in FDD, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
mode: No help available

set(*cell_name: str*, *mode: ConfigMode*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwpid>:UL:MODE
driver.configure.signaling.nradio.cell.bwp.uplink.mode.set(cell_name = 'abc',␣
↪mode = enums.ConfigMode.AUTO, bwParts = repcap.BwParts.Default)
```

Selects a configuration mode for the UL BWP settings in FDD, for BWP <bb>.

param cell_name
No help available

param mode
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.279 Rb

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:BWP<bwpid>:UL:RB
```

class RbCls

Rb commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Number_Rb: int: No parameter help available
- Start_Rb: int: No parameter help available

get(*cell_name: str*, *bwParts=BwParts.Default*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwpid>:UL:RB
value: GetStruct = driver.configure.signaling.nradio.cell.bwp.uplink.rb.
↪get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Defines the uplink of BWP <bb> in the frequency domain as a contiguous set of RBs, within the carrier bandwidth.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *number_rb: int*, *start_rb: int = None*, *bwParts=BwParts.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:BWP<bwpid>:UL:RB
driver.configure.signaling.nradio.cell.bwp.uplink.rb.set(cell_name = 'abc',␣
↪number_rb = 1, start_rb = 1, bwParts = repcap.BwParts.Default)
```

Defines the uplink of BWP <bb> in the frequency domain as a contiguous set of RBs, within the carrier bandwidth.

param cell_name
No help available

param number_rb
No help available

param start_rb
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.3.4.10.2.280 Cdrx

class CdrxCls

Cdrx commands group definition. 19 total commands, 11 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.cdrx.clone()
```

Subgroups

6.3.4.10.2.281 AaScheduler

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:CDRX:AASCheduler
```

class AaSchedulerCls

AaScheduler commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CDRX:AASCheduler
value: bool = driver.configure.signaling.nradio.cell.cdrx.aaScheduler.get(cell_
↪name = 'abc')
```

Enables or disables automatic scheduling to ensure gaps for DRX opportunities. If connected DRX is disabled, this setting is ignored (implicit OFF) . Enable connected DRX via [CONFigure:]SIGNaling:NRADio:CELL:CDRX:ENABle.

param cell_name
No help available

return
enable: - ON: The scheduler allocates DL resources only if there is DL data. Without queued DL data, no DL resources are allocated and there is an opportunity for DRX. In the UL direction, the scheduler allocates resources only upon request by the UE. - OFF: The configured scheduling applies.

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CDRX:AASCheduler
driver.configure.signaling.nradio.cell.cdrx.aaScheduler.set(cell_name = 'abc',␣
↪enable = False)
```

Enables or disables automatic scheduling to ensure gaps for DRX opportunities. If connected DRX is disabled, this setting is ignored (implicit OFF) . Enable connected DRX via [CONFigure:]SIGNaling:NRADio:CELL:CDRX:ENABle.

param cell_name
No help available

param enable

- ON: The scheduler allocates DL resources only if there is DL data. Without queued DL data, no DL resources are allocated and there is an opportunity for DRX. In the UL direction, the scheduler allocates resources only upon request by the UE.
- OFF: The configured scheduling applies.

6.3.4.10.2.282 All

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:CDRX:ALL
```

class AllCls

All commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Enable: bool: Enables or disables connected DRX.
- Od_Timer: enums.OnDurationTimer: 'drx-onDurationTimer'
- Itimer: int: 'drx-InactivityTimer'
- Soffset: int: 'drx-SlotOffset'
- Dlhr_Timer: int: 'drx-HARQ-RTT-TimerDL'
- Dlr_Timer: int: 'drx-RetransmissionTimerDL'
- Ul_Hr_Timer: int: 'drx-HARQ-RTT-TimerUL'
- Ulr_Timer: int: 'drx-RetransmissionTimerUL'
- Ld_Rx_Cycle: int: Duration of one long DRX cycle
- Ld_Rx_Start_Offset: int: 'drx-StartOffset'
- Sd_Rx_Enable: bool: Enables or disables short DRX cycles.
- Sd_Rx_Cycle: int: Duration of one short DRX cycle.
- Sd_Rx_Sc_Timer: int: Short cycle timer
- Wus_Enable: bool: Enables or disables sending DCI with format 2-6 before each long DRX cycle.
- Wus_Mode: enums.WusMode: Mode for sending wake-up signals. ON to OFF ratio or user-defined.
- Wus_Ratio_On_Off: float: Percentage of long DRX cycles for which the UE is woken up via DCI with format 2-6 containing wake-up indication = 1.

class SetStruct

Structure for setting input parameters. Contains optional setting parameters. Fields:

- Cell_Name: str: No parameter help available
- Enable: bool: Optional setting parameter. Enables or disables connected DRX.
- Od_Timer: enums.OnDurationTimer: Optional setting parameter. 'drx-onDurationTimer'
- Itimer: int: Optional setting parameter. 'drx-InactivityTimer'

- Soffset: int: Optional setting parameter. 'drx-SlotOffset'
- Dlhr_Timer: int: Optional setting parameter. 'drx-HARQ-RTT-TimerDL'
- Dlr_Timer: int: Optional setting parameter. 'drx-RetransmissionTimerDL'
- Ul_Hr_Timer: int: Optional setting parameter. 'drx-HARQ-RTT-TimerUL'
- Ulr_Timer: int: Optional setting parameter. 'drx-RetransmissionTimerUL'
- Ld_Rx_Cycle: int: Optional setting parameter. Duration of one long DRX cycle
- Ld_Rx_Start_Offset: int: Optional setting parameter. 'drx-StartOffset'
- Sd_Rx_Enable: bool: Optional setting parameter. Enables or disables short DRX cycles.
- Sd_Rx_Cycle: int: Optional setting parameter. Duration of one short DRX cycle.
- Sd_Rx_Sc_Timer: int: Optional setting parameter. Short cycle timer
- Wus_Enable: bool: Optional setting parameter. Enables or disables sending DCI with format 2-6 before each long DRX cycle.
- Wus_Mode: enums.WusMode: Optional setting parameter. Mode for sending wake-up signals. ON to OFF ratio or user-defined.
- Wus_Ratio_On_Off: float: Optional setting parameter. Percentage of long DRX cycles for which the UE is woken up via DCI with format 2-6 containing wake-up indication = 1.

get(*cell_name: str = None*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CDRX:ALL
value: GetStruct = driver.configure.signaling.nradio.cell.cdrx.all.get(cell_
↪name = 'abc')
```

Configures connected DRX settings (combination of other commands) .

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*structure: SetStruct*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CDRX:ALL
structure = driver.configure.signaling.nradio.cell.cdrx.all.SetStruct()
structure.Cell_Name: str = 'abc'
structure.Enable: bool = False
structure.Od_Timer: enums.OnDurationTimer = enums.OnDurationTimer.M1
structure.Itimer: int = 1
structure.Soffset: int = 1
structure.Dlhr_Timer: int = 1
structure.Dlr_Timer: int = 1
structure.Ul_Hr_Timer: int = 1
structure.Ulr_Timer: int = 1
structure.Ld_Rx_Cycle: int = 1
structure.Ld_Rx_Start_Offset: int = 1
structure.Sd_Rx_Enable: bool = False
structure.Sd_Rx_Cycle: int = 1
structure.Sd_Rx_Sc_Timer: int = 1
```

(continues on next page)

(continued from previous page)

```
structure.Wus_Enable: bool = False
structure.Wus_Mode: enums.WusMode = enums.WusMode.RATio
structure.Wus_Ratio_On_Off: float = 1.0
driver.configure.signaling.nradio.cell.cdrx.all.set(structure)
```

Configures connected DRX settings (combination of other commands) .

param structure
for set value, see the help for SetStruct structure arguments.

6.3.4.10.2.283 Downlink

class DownlinkCls

Downlink commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.cdrx.downlink.clone()
```

Subgroups

6.3.4.10.2.284 HrTimer

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:CDRX:DL:HRTimer
```

class HrTimerCls

HrTimer commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CDRX:DL:HRTimer
value: int = driver.configure.signaling.nradio.cell.cdrx.downlink.hrTimer.
↪get(cell_name = 'abc')
```

Configures the 'drx-HARQ-RTT-TimerDL'.

param cell_name
No help available

return
timer: No help available

set(*cell_name: str*, *timer: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CDRX:DL:HRTimer
driver.configure.signaling.nradio.cell.cdrx.downlink.hrTimer.set(cell_name =
↪'abc', timer = 1)
```

Configures the 'drx-HARQ-RTT-TimerDL'.

param cell_name
No help available

param timer
No help available

6.3.4.10.2.285 Rtimer

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:CDRX:DL:RTIMer
```

class RtimerCls

Rtimer commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CDRX:DL:RTIMer
value: int = driver.configure.signaling.nradio.cell.cdrx.downlink.rtimer.
↪get(cell_name = 'abc')
```

Configures the 'drx-RetransmissionTimerDL'.

param cell_name
No help available

return
timer: No help available

set(*cell_name: str*, *timer: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CDRX:DL:RTIMer
driver.configure.signaling.nradio.cell.cdrx.downlink.rtimer.set(cell_name = 'abc
↪', timer = 1)
```

Configures the 'drx-RetransmissionTimerDL'.

param cell_name
No help available

param timer
No help available

6.3.4.10.2.286 Enable

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:CDRX:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CDRX:ENABle
value: bool = driver.configure.signaling.nradio.cell.cdrx.enable.get(cell_name␣
↪= 'abc')
```

Enables or disables connected DRX.

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CDRX:ENABle
driver.configure.signaling.nradio.cell.cdrx.enable.set(cell_name = 'abc',␣
↪enable = False)
```

Enables or disables connected DRX.

param cell_name
No help available

param enable
No help available

6.3.4.10.2.287 Itimer

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:CDRX:ITIMer
```

class ItimerCls

Itimer commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CDRX:ITIMer
value: int = driver.configure.signaling.nradio.cell.cdrx.itimer.get(cell_name =
↪'abc')
```

Configures the 'drx-InactivityTimer'.

param cell_name
No help available

return
timer: No help available

set(*cell_name: str*, *timer: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CDRX:ITIMer
driver.configure.signaling.nradio.cell.cdrx.itimer.set(cell_name = 'abc', timer␣
↪= 1)
```

Configures the 'drx-InactivityTimer'.

param cell_name
No help available

param timer
No help available

6.3.4.10.2.288 Ldrx

class LdrxCls

Ldrx commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.cdrx.ldrx.clone()
```

Subgroups

6.3.4.10.2.289 Cycle

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:CDRX:LDRX:CYCLe
```

class CycleCls

Cycle commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CDRX:LDRX:CYCLe
value: int = driver.configure.signaling.nradio.cell.cdrx.ldrx.cycle.get(cell_
↪name = 'abc')
```

Configures the duration of one long DRX cycle. The long DRX cycle duration must be divisible by the short DRX cycle duration.

param cell_name
No help available

return
cycle: No help available

set(*cell_name: str*, *cycle: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CDRX:LDRX:CYCLe
driver.configure.signaling.nradio.cell.cdrx.ldrx.cycle.set(cell_name = 'abc',␣
↪cycle = 1)
```

Configures the duration of one long DRX cycle. The long DRX cycle duration must be divisible by the short DRX cycle duration.

param cell_name
No help available

param cycle
No help available

6.3.4.10.2.290 Soffset

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:CDRX:LDRX:SOFFset
```

class SoffsetCls

Soffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CDRX:LDRX:SOFFset
value: int = driver.configure.signaling.nradio.cell.cdrx.ldrx.soffset.get(cell_
↪name = 'abc')
```

Configures the drx-StartOffset, shifting the DRX cycle start.

param cell_name
No help available

return
start_offset: No help available

set(*cell_name: str*, *start_offset: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CDRX:LDRX:SOFFset
driver.configure.signaling.nradio.cell.cdrx.ldrx.soffset.set(cell_name = 'abc',␣
↪start_offset = 1)
```

Configures the drx-StartOffset, shifting the DRX cycle start.

param cell_name
No help available

param start_offset
No help available

6.3.4.10.2.291 OdTimer

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:CDRX:ODTimer
```

class OdTimerCls

OdTimer commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → OnDurationTimer

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CDRX:ODTimer
value: enums.OnDurationTimer = driver.configure.signaling.nradio.cell.cdrx.
↪odTimer.get(cell_name = 'abc')
```

Configures the 'drx-onDurationTimer'.

param cell_name
No help available

return
timer: M1D to M31D: 1/32 ms to 31/32 ms M1 to M800: 1 ms to 800 ms M1K0 to M1K6: 1000 ms to 1600 ms

set(*cell_name: str*, *timer: OnDurationTimer*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CDRX:ODTimer
driver.configure.signaling.nradio.cell.cdrx.odTimer.set(cell_name = 'abc',␣
↪timer = enums.OnDurationTimer.M1)
```

Configures the 'drx-onDurationTimer'.

param cell_name
No help available

param timer
M1D to M31D: 1/32 ms to 31/32 ms M1 to M800: 1 ms to 800 ms M1K0 to M1K6: 1000 ms to 1600 ms

6.3.4.10.2.292 Sdrx

class SdrxCls

Sdrx commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.cdrx.sdrx.clone()
```

Subgroups

6.3.4.10.2.293 Cycle

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:CDRX:SDRX:CYCLe
```

class CycleCls

Cycle commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CDRX:SDRX:CYCLe
value: int = driver.configure.signaling.nradio.cell.cdrx.sdrx.cycle.get(cell_
↪name = 'abc')
```

Configures the duration of one short DRX cycle. The long DRX cycle duration must be divisible by the short DRX cycle duration.

param cell_name
No help available

return
cycle: No help available

set(*cell_name: str*, *cycle: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CDRX:SDRX:CYCLe
driver.configure.signaling.nradio.cell.cdrx.sdrx.cycle.set(cell_name = 'abc',␣
↪cycle = 1)
```

Configures the duration of one short DRX cycle. The long DRX cycle duration must be divisible by the short DRX cycle duration.

param cell_name
No help available

param cycle
No help available

6.3.4.10.2.294 Enable

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:CDRX:SDRX:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CDRX:SDRX:ENABle
value: bool = driver.configure.signaling.nradio.cell.cdrx.sdrx.enable.get(cell_
↪name = 'abc')
```

Enables or disables short DRX cycles.

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CDRX:SDRX:ENABle
driver.configure.signaling.nradio.cell.cdrx.sdrx.enable.set(cell_name = 'abc',␣
↪enable = False)
```

Enables or disables short DRX cycles.

param cell_name
No help available

param enable
No help available

6.3.4.10.2.295 ScTimer

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:CDRX:SDRX:SCTimer
```

class ScTimerCls

ScTimer commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CDRX:SDRX:SCTimer
value: int = driver.configure.signaling.nradio.cell.cdrx.sdrx.scTimer.get(cell_
↪name = 'abc')
```

Configures the short cycle timer.

param cell_name
No help available

return
timer: No help available

set(*cell_name: str*, *timer: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CDRX:SDRX:SCTimer
driver.configure.signaling.nradio.cell.cdrx.sdrx.scTimer.set(cell_name = 'abc',␣
↪timer = 1)
```

Configures the short cycle timer.

param cell_name
No help available

param timer
No help available

6.3.4.10.2.296 Soffset

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:CDRX:SOFFset
```

class SoffsetCls

Soffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CDRX:SOFFset
value: int = driver.configure.signaling.nradio.cell.cdrx.soffset.get(cell_name␣
↪= 'abc')
```

Configures the 'drx-SlotOffset'.

param cell_name
No help available

return
offset: No help available

set(*cell_name: str*, *offset: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CDRX:SOFFset
driver.configure.signaling.nradio.cell.cdrx.soffset.set(cell_name = 'abc',␣
↪offset = 1)
```

Configures the 'drx-SlotOffset'.

param cell_name
No help available

param offset
No help available

6.3.4.10.2.297 Uplink

class UplinkCls

Uplink commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.cdrx.uplink.clone()
```

Subgroups

6.3.4.10.2.298 HrTimer

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:CDRX:UL:HRTimer
```

class HrTimerCls

HrTimer commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CDRX:UL:HRTimer
value: int = driver.configure.signaling.nradio.cell.cdrx.uplink.hrTimer.
↪get(cell_name = 'abc')
```

Configures the 'drx-HARQ-RTT-TimerUL'.

param cell_name
No help available

return
timer: No help available

set(*cell_name: str*, *timer: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CDRX:UL:HRTimer
driver.configure.signaling.nradio.cell.cdrx.uplink.hrTimer.set(cell_name = 'abc
↪', timer = 1)
```

Configures the 'drx-HARQ-RTT-TimerUL'.

param cell_name
No help available

param timer
No help available

6.3.4.10.2.299 Rtimer

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:CDRX:UL:RTIMer
```

class RtimerCls

Rtimer commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CDRX:UL:RTIMer
value: int = driver.configure.signaling.nradio.cell.cdrx.uplink.rtimer.get(cell_
↪name = 'abc')
```

Configures the 'drx-RetransmissionTimerUL'.

param cell_name
No help available

return
timer: No help available

set(*cell_name: str*, *timer: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CDRX:UL:RTIMer
driver.configure.signaling.nradio.cell.cdrx.uplink.rtimer.set(cell_name = 'abc',
↪ timer = 1)
```

Configures the 'drx-RetransmissionTimerUL'.

param cell_name
No help available

param timer
No help available

6.3.4.10.2.300 Wus

class WusCls

Wus commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.cdrx.wus.clone()
```

Subgroups

6.3.4.10.2.301 All

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:CDRX:WUS:ALL
```

class AllCls

All commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Enable: bool: Enables sending DCI with format 2-6 before each long DRX cycle.
- Mode: enums.WusMode: Mode for sending wake-up signals. ON to OFF ratio or user-defined.
- Ratio_On_Off: float: Percentage of long DRX cycles for which the UE is woken up via DCI with format 2-6 containing wake-up indication = 1.

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CDRX:WUS:ALL
value: GetStruct = driver.configure.signaling.nradio.cell.cdrx.wus.all.get(cell_
↪name = 'abc')
```

Configures connected DRX wake-up signal settings (combination of other commands) .

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *enable: bool*, *mode: WusMode = None*, *ratio_on_off: float = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CDRX:WUS:ALL
driver.configure.signaling.nradio.cell.cdrx.wus.all.set(cell_name = 'abc',␣
↪enable = False, mode = enums.WusMode.RATio, ratio_on_off = 1.0)
```

Configures connected DRX wake-up signal settings (combination of other commands) .

param cell_name
No help available

param enable
Enables sending DCI with format 2-6 before each long DRX cycle.

param mode
Mode for sending wake-up signals. ON to OFF ratio or user-defined.

param ratio_on_off
Percentage of long DRX cycles for which the UE is woken up via DCI with format 2-6 containing wake-up indication = 1.

6.3.4.10.2.302 Enable

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:CDRX:WUS:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CDRX:WUS:ENABle
value: bool = driver.configure.signaling.nradio.cell.cdrx.wus.enable.get(cell_
↪name = 'abc')
```

Enables or disables sending DCI with format 2-6 before each long DRX cycle.

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CDRX:WUS:ENABle
driver.configure.signaling.nradio.cell.cdrx.wus.enable.set(cell_name = 'abc',␣
↪enable = False)
```

Enables or disables sending DCI with format 2-6 before each long DRX cycle.

param cell_name
No help available

param enable
No help available

6.3.4.10.2.303 Mode

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:CDRX:WUS:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → WusMode

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CDRX:WUS:MODE
value: enums.WusMode = driver.configure.signaling.nradio.cell.cdrx.wus.mode.
↪get(cell_name = 'abc')
```

No command help available

param cell_name
No help available

return
mode: No help available

set(*cell_name: str*, *mode: WusMode*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CDRX:WUS:MODE
driver.configure.signaling.nradio.cell.cdrx.wus.mode.set(cell_name = 'abc',␣
↪mode = enums.WusMode.RATio)
```

No command help available

param cell_name
No help available

param mode
No help available

6.3.4.10.2.304 Ratio

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:CDRX:WUS:RATio
```

class RatioCls

Ratio commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CDRX:WUS:RATio
value: float = driver.configure.signaling.nradio.cell.cdrx.wus.ratio.get(cell_
↪name = 'abc')
```

Configures the percentage of long DRX cycles for which the UE is woken up via DCI with format 2-6 containing wake-up indication = 1.

param cell_name
No help available

return
ratio_on_off: No help available

set(*cell_name: str*, *ratio_on_off: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CDRX:WUS:RATio
driver.configure.signaling.nradio.cell.cdrx.wus.ratio.set(cell_name = 'abc',␣
↪ratio_on_off = 1.0)
```

Configures the percentage of long DRX cycles for which the UE is woken up via DCI with format 2-6 containing wake-up indication = 1.

param cell_name
No help available

param ratio_on_off
No help available

6.3.4.10.2.305 Cmatrix

class CmatrixCls

Cmatrix commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.cmatrix.clone()
```

Subgroups

6.3.4.10.2.306 Mode

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:CMATrix:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → ModeFrecoveryB

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CMATrix:MODE
value: enums.ModeFrecoveryB = driver.configure.signaling.nradio.cell.cmatrix.
↪mode.get(cell_name = 'abc')
```

Activates a channel matrix.

param cell_name
No help available

return
mode: OFF: Without a fader in the signal path, IDENtity matrix. With a fader in the signal path, TGPP matrix. UDEFined: Matrix defined via another operating interface. TGPP: Matrix defined in 3GPP TS 38.101-4 annex B.1. HADamard: Hadamard matrix. IDENtity: Identity matrix.

set(*cell_name: str*, *mode: ModeFrecoveryB*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CMATrix:MODE
driver.configure.signaling.nradio.cell.cmatrix.mode.set(cell_name = 'abc', mode␣
↪= enums.ModeFrecoveryB.HADamard)
```

Activates a channel matrix.

param cell_name
No help available

param mode
OFF: Without a fader in the signal path, IDENtity matrix. With a fader in the signal path, TGPP matrix. UDEFined: Matrix defined via another operating interface. TGPP: Matrix defined in 3GPP TS 38.101-4 annex B.1. HADamard: Hadamard matrix. IDENtity: Identity matrix.

6.3.4.10.2.307 CqiReporting

class CqiReportingCls

CqiReporting commands group definition. 10 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.cqiReporting.clone()
```

Subgroups

6.3.4.10.2.308 Combined

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:CQIReporting:COMBined
```

class CombinedCls

Combined commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Enable: enums.EnableCqi: Selects the CSI reporting type. OFF: no reporting PERiodic: periodic CSI reporting APERiodic: aperiodic CSI reporting SPERsistant: semi-persistent CSI reporting
- Periodicity: enums.PeriodicityCqiReport: Global periodicity for CSI reporting and CSI-RS resources, in slots.
- Offset: int: Offset value of 'periodicityAndOffset' in IE 'NZP-CSI-RS-Resource'. The offset must be less than the periodicity.
- Ports: enums.Ports: The number of CSI-RS ports that is signaled as 'nrofPorts' in IE 'CSI-RS-ResourceMapping'.
- Symbol: int: The first OFDM symbol in the RB used for CSI-RS.
- Power: enums.RsrcPower: Power offset of NZP CSI-RS RE to SSS RE. -9 dB, -6 dB, -3 dB, 0 dB, +3 dB, +6 dB
- Report_Offset: int: Offset value of 'reportSlotConfig'. The offset must be less than the periodicity.
- Report_Cqi: enums.ReportCqi: 'cqi-FormatIndicator' signaled to the UE. OFF: no CQI reporting WB: wideband CQI reporting SB: subband CQI reporting
- Report_Pmi: enums.ReportCqi: 'pmi-FormatIndicator' signaled to the UE. OFF: no PMI reporting WB: wideband PMI reporting SB: subband PMI reporting

class SetStruct

Structure for setting input parameters. Contains optional setting parameters. Fields:

- Cell_Name: str: No parameter help available
- Enable: enums.EnableCqi: Optional setting parameter. Selects the CSI reporting type. OFF: no reporting PERiodic: periodic CSI reporting APERiodic: aperiodic CSI reporting SPERsistant: semi-persistent CSI reporting
- Periodicity: enums.PeriodicityCqiReport: Optional setting parameter. Global periodicity for CSI reporting and CSI-RS resources, in slots.
- Offset: int: Optional setting parameter. Offset value of 'periodicityAndOffset' in IE 'NZP-CSI-RS-Resource'. The offset must be less than the periodicity.

- Ports: enums.Ports: Optional setting parameter. The number of CSI-RS ports that is signaled as 'nrofPorts' in IE 'CSI-RS-ResourceMapping'.
- Symbol: int: Optional setting parameter. The first OFDM symbol in the RB used for CSI-RS.
- Power: enums.RsrcPower: Optional setting parameter. Power offset of NZP CSI-RS RE to SSS RE. -9 dB, -6 dB, -3 dB, 0 dB, +3 dB, +6 dB
- Report_Offset: int: Optional setting parameter. Offset value of 'reportSlotConfig'. The offset must be less than the periodicity.
- Report_Cqi: enums.ReportCqi: Optional setting parameter. 'cqi-FormatIndicator' signaled to the UE. OFF: no CQI reporting WB: wideband CQI reporting SB: subband CQI reporting
- Report_Pmi: enums.ReportCqi: Optional setting parameter. 'pmi-FormatIndicator' signaled to the UE. OFF: no PMI reporting WB: wideband PMI reporting SB: subband PMI reporting

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CQIReporting:COMBined
value: GetStruct = driver.configure.signaling.nradio.cell.cqiReporting.combined.
↪get(cell_name = 'abc')
```

Configures several CQI reporting settings simultaneously.

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*structure: SetStruct*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CQIReporting:COMBined
structure = driver.configure.signaling.nradio.cell.cqiReporting.combined.
↪SetStruct()
structure.Cell_Name: str = 'abc'
structure.Enable: enums.EnableCqi = enums.EnableCqi.APERiodic
structure.Periodicity: enums.PeriodicityCqiReport = enums.PeriodicityCqiReport.
↪P10
structure.Offset: int = 1
structure.Ports: enums.Ports = enums.Ports.P1
structure.Symbol: int = 1
structure.Power: enums.RsrcPower = enums.RsrcPower.M3DB
structure.Report_Offset: int = 1
structure.Report_Cqi: enums.ReportCqi = enums.ReportCqi.OFF
structure.Report_Pmi: enums.ReportCqi = enums.ReportCqi.OFF
driver.configure.signaling.nradio.cell.cqiReporting.combined.set(structure)
```

Configures several CQI reporting settings simultaneously.

param structure
for set value, see the help for SetStruct structure arguments.

6.3.4.10.2.309 Enable

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:CQIReporting:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → EnableCqi

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CQIReporting:ENABle
value: enums.EnableCqi = driver.configure.signaling.nradio.cell.cqiReporting.
↪enable.get(cell_name = 'abc')
```

Selects the CSI reporting type.

param cell_name
No help available

return
enable: OFF: no reporting PERiodic: periodic CSI reporting APERiodic: aperiodic CSI reporting SPERsistant: semi-persistent CSI reporting

set(*cell_name: str*, *enable: EnableCqi*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CQIReporting:ENABle
driver.configure.signaling.nradio.cell.cqiReporting.enable.set(cell_name = 'abc
↪', enable = enums.EnableCqi.APERiodic)
```

Selects the CSI reporting type.

param cell_name
No help available

param enable
OFF: no reporting PERiodic: periodic CSI reporting APERiodic: aperiodic CSI reporting SPERsistant: semi-persistent CSI reporting

6.3.4.10.2.310 Periodicity

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:CQIReporting:PERiodicity
```

class PeriodicityCls

Periodicity commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → PeriodicityCqiReport

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CQIReporting:PERiodicity
value: enums.PeriodicityCqiReport = driver.configure.signaling.nradio.cell.
↪cqiReporting.periodicity.get(cell_name = 'abc')
```

Selects the global periodicity for CSI reporting and CSI-RS resources.

param cell_name
No help available

return
periodicity: Periodicity in slots

set(*cell_name: str*, *periodicity: PeriodicityCqiReport*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CQIReporting:PERiodicity
driver.configure.signaling.nradio.cell.cqiReporting.periodicity.set(cell_name =
↪'abc', periodicity = enums.PeriodicityCqiReport.P10)
```

Selects the global periodicity for CSI reporting and CSI-RS resources.

param cell_name
No help available

param periodicity
Periodicity in slots

6.3.4.10.2.311 Report

class ReportCls

Report commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.cqiReporting.report.clone()
```

Subgroups

6.3.4.10.2.312 Cqi

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:CQIReporting:REPort:CQI
```

class CqiCls

Cqi commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → ReportCqi

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CQIReporting:REPort:CQI
value: enums.ReportCqi = driver.configure.signaling.nradio.cell.cqiReporting.
↪report.cqi.get(cell_name = 'abc')
```

Configures the parameter 'cqi-FormatIndicator' signaled to the UE.

param cell_name
No help available

return
report_format: OFF: no CQI reporting WB: wideband CQI reporting SB: subband CQI reporting

set(*cell_name: str*, *report_format: ReportCqi*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CQIReporting:REPort:CQI
driver.configure.signaling.nradio.cell.cqiReporting.report.cqi.set(cell_name =
↪'abc', report_format = enums.ReportCqi.OFF)
```

Configures the parameter 'cqi-FormatIndicator' signaled to the UE.

param cell_name
No help available

param report_format
OFF: no CQI reporting WB: wideband CQI reporting SB: subband CQI reporting

6.3.4.10.2.313 Offset

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:CQIReporting:REPort:OFFSet
```

class OffsetCls

Offset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CQIReporting:REPort:OFFSet
value: int = driver.configure.signaling.nradio.cell.cqiReporting.report.offset.
↪get(cell_name = 'abc')
```

Configures the offset value of 'reportSlotConfig'. The offset must be less than the periodicity, see method **RsCMX_Signaling.Sense.Signaling.Nradio.Cell.CqiReporting.Report.Periodicity.get_**.

param cell_name
No help available

return
offset: No help available

set(*cell_name: str*, *offset: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CQIReporting:REPort:OFFSet
driver.configure.signaling.nradio.cell.cqiReporting.report.offset.set(cell_name␣
↪= 'abc', offset = 1)
```

Configures the offset value of 'reportSlotConfig'. The offset must be less than the periodicity, see method **RsCMX_Signaling.Sense.Signaling.Nradio.Cell.CqiReporting.Report.Periodicity.get_**.

param cell_name
No help available

param offset
No help available

6.3.4.10.2.314 Pmi

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:CQIReporting:REPort:PMI
```

class PmiCls

Pmi commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → ReportCqi

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CQIReporting:REPort:PMI
value: enums.ReportCqi = driver.configure.signaling.nradio.cell.cqiReporting.
↪report.pmi.get(cell_name = 'abc')
```

Configures the parameter 'pmi-FormatIndicator' signaled to the UE.

param cell_name
No help available

return
report_format: OFF: no PMI reporting WB: wideband PMI reporting SB: subband PMI reporting

set(*cell_name: str*, *report_format: ReportCqi*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CQIReporting:REPort:PMI
driver.configure.signaling.nradio.cell.cqiReporting.report.pmi.set(cell_name =
↪'abc', report_format = enums.ReportCqi.OFF)
```

Configures the parameter 'pmi-FormatIndicator' signaled to the UE.

param cell_name
No help available

param report_format
OFF: no PMI reporting WB: wideband PMI reporting SB: subband PMI reporting

6.3.4.10.2.315 Resource

class ResourceCls

Resource commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.cqiReporting.resource.clone()
```

Subgroups

6.3.4.10.2.316 FoSymbol

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:CQIReporting:RESource:FOSYmbol
```

class FoSymbolCls

FoSymbol commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CQIReporting:RESource:FOSYmbol
value: int = driver.configure.signaling.nradio.cell.cqiReporting.resource.
↪foSymbol.get(cell_name = 'abc')
```

Configures the first OFDM symbol in the RB used for CSI-RS.

param cell_name
No help available

return
symbol: No help available

set(*cell_name: str*, *symbol: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CQIReporting:RESource:FOSYmbol
driver.configure.signaling.nradio.cell.cqiReporting.resource.foSymbol.set(cell_
↪name = 'abc', symbol = 1)
```

Configures the first OFDM symbol in the RB used for CSI-RS.

param cell_name
No help available

param symbol
No help available

6.3.4.10.2.317 Offset

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:CQIReporting:RESource:OFFSet
```

class OffsetCls

Offset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CQIReporting:RESource:OFFSet
value: int = driver.configure.signaling.nradio.cell.cqiReporting.resource.
↪offset.get(cell_name = 'abc')
```

Configures the offset value of 'periodicityAndOffset' in IE 'NZP-CSI-RS-Resource'. The offset must be less than the periodicity, see method RsCMX_Signaling.Sense.Signaling.Nradio.Cell.CqiReporting.Resource.Periodicity.get_.

param cell_name
No help available

return
offset: No help available

set(*cell_name: str*, *offset: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CQIReporting:RESource:OFFSet
driver.configure.signaling.nradio.cell.cqiReporting.resource.offset.set(cell_
↪name = 'abc', offset = 1)
```

Configures the offset value of 'periodicityAndOffset' in IE 'NZP-CSI-RS-Resource'. The offset must be less than the periodicity, see method RsCMX_Signaling.Sense.Signaling.Nradio.Cell.CqiReporting.Resource.Periodicity.get_.

param cell_name
No help available

param offset
No help available

6.3.4.10.2.318 Ports

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:CQIReporting:RESource:PORTs
```

class PortsCls

Ports commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → Ports

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CQIReporting:RESource:PORTs
value: enums.Ports = driver.configure.signaling.nradio.cell.cqiReporting.
↪resource.ports.get(cell_name = 'abc')
```

Selects the number of CSI-RS ports, signaled as 'nrofPorts' in IE 'CSI-RS-ResourceMapping'.

param cell_name
No help available

return
ports: No help available

set(*cell_name: str*, *ports: Ports*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CQIReporting:RESource:PORTs
driver.configure.signaling.nradio.cell.cqiReporting.resource.ports.set(cell_
↪name = 'abc', ports = enums.Ports.P1)
```

Selects the number of CSI-RS ports, signaled as 'nrofPorts' in IE 'CSI-RS-ResourceMapping'.

param cell_name
No help available

param ports
No help available

6.3.4.10.2.319 PoVsss

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:CQIReporting:RESource:POVSss
```

class PoVsssCls

PoVsss commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → RsrcPower

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CQIReporting:RESource:POVSss
value: enums.RsrcPower = driver.configure.signaling.nradio.cell.cqiReporting.
↪resource.poVsss.get(cell_name = 'abc')
```

Configures the power offset of NZP CSI-RS RE to SSS RE.

param cell_name
No help available

return
power: -9 dB, -6 dB, -3 dB, 0 dB, +3 dB, +6 dB

set(*cell_name: str*, *power: RsrcPower*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CQIReporting:RESource:POVSss
driver.configure.signaling.nradio.cell.cqiReporting.resource.poVsss.set(cell_
↪name = 'abc', power = enums.RsrcPower.M3DB)
```

Configures the power offset of NZP CSI-RS RE to SSS RE.

param cell_name
No help available

param power
-9 dB, -6 dB, -3 dB, 0 dB, +3 dB, +6 dB

6.3.4.10.2.320 Csi

class CsiCls

Csi commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.csi.clone()
```

Subgroups

6.3.4.10.2.321 Trs

class TrsCls

Trs commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.csi.trs.clone()
```

Subgroups

6.3.4.10.2.322 Config

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:CSI:TRS:CONFig
```

class ConfigCls

Config commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Index: int: Number of the TRS configuration.
- Bw_Selection: enums.BwSelection: All RBs of the BWP or maximum 52 RBs.
- Slot_Offset: int: Time domain offset.
- Symbol_Pair: enums.SymbolPair: Selects the two OFDM symbols used for TRS. The first digit indicates the first symbol. The remaining digits indicate the second symbol. Example: S913 means symbol 9 and symbol 13.
- Periodicity: enums.TrsPeriodicity: Periodicity for transmission of the resource set, in ms.
- No_Consec_Slots: int: Number of slots per resource set in FR2.

class SetStruct

Structure for setting input parameters. Fields:

- Cell_Name: str: No parameter help available
- Index: int: Number of the TRS configuration.
- Bw_Selection: enums.BwSelection: All RBs of the BWP or maximum 52 RBs.

- Slot_Offset: int: Time domain offset.
- Symbol_Pair: enums.SymbolPair: Selects the two OFDM symbols used for TRS. The first digit indicates the first symbol. The remaining digits indicate the second symbol. Example: S913 means symbol 9 and symbol 13.
- Periodicity: enums.TrsPperiodicity: Periodicity for transmission of the resource set, in ms.
- No_Consec_Slots: int: Number of slots per resource set in FR2.

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CSI:TRS:CONFig
value: GetStruct = driver.configure.signaling.nradio.cell.csi.trs.config.
↪get(cell_name = 'abc')
```

Defines settings of TRS <Index>, for the initial BWP. If there are several TRS configurations, a query returns the settings of all TRS configurations.

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*structure: SetStruct*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CSI:TRS:CONFig
structure = driver.configure.signaling.nradio.cell.csi.trs.config.SetStruct()
structure.Cell_Name: str = 'abc'
structure.Index: int = 1
structure.Bw_Selection: enums.BwSelection = enums.BwSelection.ALL
structure.Slot_Offset: int = 1
structure.Symbol_Pair: enums.SymbolPair = enums.SymbolPair.S04
structure.Periodicity: enums.TrsPeriodicity = enums.TrsPeriodicity.P10
structure.No_Consec_Slots: int = 1
driver.configure.signaling.nradio.cell.csi.trs.config.set(structure)
```

Defines settings of TRS <Index>, for the initial BWP. If there are several TRS configurations, a query returns the settings of all TRS configurations.

param structure
for set value, see the help for SetStruct structure arguments.

6.3.4.10.2.323 Mode

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:CSI:TRS:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → ModeTrs

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CSI:TRS:MODE
value: enums.ModeTrs = driver.configure.signaling.nradio.cell.csi.trs.mode.
↪get(cell_name = 'abc')
```

Selects the configuration mode for TRS transmission, for the initial BWP.

param cell_name
No help available

return
mode: OFF: no TRS DEF: TRS according to 3GPP TS 38.508 UDEF: user-defined TRS

set(*cell_name: str*, *mode: ModeTrs*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CSI:TRS:MODE
driver.configure.signaling.nradio.cell.csi.trs.mode.set(cell_name = 'abc', mode␣
↪= enums.ModeTrs.DEF)
```

Selects the configuration mode for TRS transmission, for the initial BWP.

param cell_name
No help available

param mode
OFF: no TRS DEF: TRS according to 3GPP TS 38.508 UDEF: user-defined TRS

6.3.4.10.2.324 CssZero

class CssZeroCls

CssZero commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.cssZero.clone()
```

Subgroups

6.3.4.10.2.325 CrZero

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:CSSZero:CRZero
```

class CrZeroCls

CrZero commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CSSZero:CRZero
value: int = driver.configure.signaling.nradio.cell.cssZero.crZero.get(cell_
↪name = 'abc')
```

Specifies the common control resource set (CORESET) number 0.

param cell_name
No help available

return
zero: No help available

set(*cell_name: str*, *zero: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CSSZero:CRZero
driver.configure.signaling.nradio.cell.cssZero.crZero.set(cell_name = 'abc',␣
↪zero = 1)
```

Specifies the common control resource set (CORESET) number 0.

param cell_name
No help available

param zero
No help available

6.3.4.10.2.326 SsZero

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:CSSZero:SSZero
```

class SsZeroCls

SsZero commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:CSSZero:SSZero
value: int = driver.configure.signaling.nradio.cell.cssZero.ssZero.get(cell_
↪name = 'abc')
```

Queries the common search space number 0.

param cell_name
No help available

return
zero: No help available

6.3.4.10.2.327 Dmrs

class DmrsCls

Dmrs commands group definition. 21 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.dmrs.clone()
```

Subgroups

6.3.4.10.2.328 Downlink

class DownlinkCls

Downlink commands group definition. 10 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.dmrs.downlink.clone()
```

Subgroups

6.3.4.10.2.329 Mta

class MtaCls

Mta commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.dmrs.downlink.mta.clone()
```

Subgroups

6.3.4.10.2.330 Length

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:DMRS:DL:MTA:LENGth
```

class LengthCls

Length commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → MaxLength

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DMRS:DL:MTA:LENGth
value: enums.MaxLength = driver.configure.signaling.nradio.cell.dmrs.downlink.
↪mta.length.get(cell_name = 'abc')
```

No command help available

param cell_name
No help available

return
max_length: No help available

set(*cell_name: str*, *max_length: MaxLength*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DMRS:DL:MTA:LENGth
driver.configure.signaling.nradio.cell.dmrs.downlink.mta.length.set(cell_name =
↪'abc', max_length = enums.MaxLength.L1)
```

No command help available

param cell_name
No help available

param max_length
No help available

6.3.4.10.2.331 Papr

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:DMRS:DL:MTA:PAPR
```

class PaprCls

Papr commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DMRS:DL:MTA:PAPR
value: bool = driver.configure.signaling.nradio.cell.dmrs.downlink.mta.papr.
↪get(cell_name = 'abc')
```

Enables the usage of a DMRS with a low PAPR, for PDSCH, mapping type A, initial BWP.

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DMRS:DL:MTA:PAPR
driver.configure.signaling.nradio.cell.dmrs.downlink.mta.papr.set(cell_name =
↪'abc', enable = False)
```

Enables the usage of a DMRS with a low PAPR, for PDSCH, mapping type A, initial BWP.

param cell_name
No help available

param enable
No help available

6.3.4.10.2.332 Position

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:DMRS:DL:MTA:POSition
```

class PositionCls

Position commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → MtxPosition

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DMRS:DL:MTA:POSition
value: enums.MtxPosition = driver.configure.signaling.nradio.cell.dmrs.downlink.
↪mta.position.get(cell_name = 'abc')
```

Defines parameter 'dmrs-AdditionalPosition' for PDSCH, mapping type A, initial BWP.

param cell_name
No help available

return
position: No help available

set(*cell_name: str*, *position: MtxPosition*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DMRS:DL:MTA:POSition
driver.configure.signaling.nradio.cell.dmrs.downlink.mta.position.set(cell_name␣
↪= 'abc', position = enums.MtxPosition.P0)
```

Defines parameter 'dmrs-AdditionalPosition' for PDSCH, mapping type A, initial BWP.

param cell_name
No help available

param position
No help available

6.3.4.10.2.333 TypePy

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:DMRS:DL:MTA:TYPE
```

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → ConfigType

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DMRS:DL:MTA:TYPE
value: enums.ConfigType = driver.configure.signaling.nradio.cell.dmrs.downlink.
↪mta.typePy.get(cell_name = 'abc')
```

No command help available

param cell_name
No help available

return
config_type: No help available

set(*cell_name: str*, *config_type: ConfigType*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DMRS:DL:MTA:TYPE
driver.configure.signaling.nradio.cell.dmrs.downlink.mta.typePy.set(cell_name =
↪'abc', config_type = enums.ConfigType.T1)
```

No command help available

param cell_name
No help available

param config_type
No help available

6.3.4.10.2.334 Mtb

class MtbCls

Mtb commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.dmrs.downlink.mtb.clone()
```

Subgroups

6.3.4.10.2.335 Length

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:DMRS:DL:MTB:LENGth
```

class LengthCls

Length commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → MaxLength

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DMRS:DL:MTB:LENGth
value: enums.MaxLength = driver.configure.signaling.nradio.cell.dmrs.downlink.
↪mtb.length.get(cell_name = 'abc')
```

No command help available

param cell_name
No help available

return
max_length: No help available

set(*cell_name: str*, *max_length: MaxLength*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DMRS:DL:MTB:LENGth
driver.configure.signaling.nradio.cell.dmrs.downlink.mtb.length.set(cell_name =
↪'abc', max_length = enums.MaxLength.L1)
```

No command help available

param cell_name
No help available

param max_length
No help available

6.3.4.10.2.336 Papr

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:DMRS:DL:MTB:PAPR
```

class PaprCls

Papr commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DMRS:DL:MTB:PAPR
value: bool = driver.configure.signaling.nradio.cell.dmrs.downlink.mtb.papr.
↪get(cell_name = 'abc')
```

Enables the usage of a DMRS with a low PAPR, for PDSCH, mapping type B, initial BWP.

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DMRS:DL:MTB:PAPR
driver.configure.signaling.nradio.cell.dmrs.downlink.mtb.papr.set(cell_name =
↪'abc', enable = False)
```

Enables the usage of a DMRS with a low PAPR, for PDSCH, mapping type B, initial BWP.

param cell_name
No help available

param enable
No help available

6.3.4.10.2.337 Position

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:DMRS:DL:MTB:POSition
```

class PositionCls

Position commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → MtxPosition

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DMRS:DL:MTB:POSition
value: enums.MtxPosition = driver.configure.signaling.nradio.cell.dmrs.downlink.
↪mtb.position.get(cell_name = 'abc')
```

Defines parameter 'dmrs-AdditionalPosition' for PDSCH, mapping type B, initial BWP.

param cell_name
No help available

return
position: No help available

set(*cell_name: str*, *position: MtxPosition*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DMRS:DL:MTB:POSition
driver.configure.signaling.nradio.cell.dmrs.downlink.mtb.position.set(cell_name␣
↪= 'abc', position = enums.MtxPosition.P0)
```

Defines parameter 'dmrs-AdditionalPosition' for PDSCH, mapping type B, initial BWP.

param cell_name
No help available

param position
No help available

6.3.4.10.2.338 TypePy

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:DMRS:DL:MTB:TYPE
```

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → ConfigType

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DMRS:DL:MTB:TYPE
value: enums.ConfigType = driver.configure.signaling.nradio.cell.dmrs.downlink.
↪mtb.typePy.get(cell_name = 'abc')
```

No command help available

param cell_name
No help available

return
config_type: No help available

set(*cell_name: str*, *config_type: ConfigType*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DMRS:DL:MTB:TYPE
driver.configure.signaling.nradio.cell.dmrs.downlink.mtb.typePy.set(cell_name =
↪'abc', config_type = enums.ConfigType.T1)
```

No command help available

param cell_name
No help available

param config_type
No help available

6.3.4.10.2.339 Ptrs

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:DMRS:DL:PTRS
```

class PtrsCls

Ptrs commands group definition. 2 total commands, 1 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Time_Dens_Pres: enums.DensityPreset: Presence of the 'timeDensity' (field signaled or not) .
- Time_Dens_Mcs_1: int: Signaled 'timeDensity', ptrs-MCS1.
- Time_Dens_Mcs_2: int: Signaled 'timeDensity', ptrs-MCS2.
- Time_Dens_Mcs_3: int: Signaled 'timeDensity', ptrs-MCS3.
- Freq_Dens_Pres: enums.DensityPreset: Presence of the 'frequencyDensity' (field signaled or not) .
- Freq_Dens_Nrb_0: int: Signaled 'frequencyDensity', NRB0.
- Freq_Dens_Nrb_1: int: Signaled 'frequencyDensity', NRB1.
- Epre_Ratio: enums.EpreRatio: Signaled 'epre-Ratio', PTRS relative to PDSCH.
- Resource_Offset: enums.ResourceOffset: Signaled 'resourceElementOffset'.

class SetStruct

Structure for setting input parameters. Fields:

- Cell_Name: str: No parameter help available
- Time_Dens_Pres: enums.DensityPreset: Presence of the 'timeDensity' (field signaled or not) .
- Time_Dens_Mcs_1: int: Signaled 'timeDensity', ptrs-MCS1.
- Time_Dens_Mcs_2: int: Signaled 'timeDensity', ptrs-MCS2.
- Time_Dens_Mcs_3: int: Signaled 'timeDensity', ptrs-MCS3.
- Freq_Dens_Pres: enums.DensityPreset: Presence of the 'frequencyDensity' (field signaled or not) .
- Freq_Dens_Nrb_0: int: Signaled 'frequencyDensity', NRB0.
- Freq_Dens_Nrb_1: int: Signaled 'frequencyDensity', NRB1.
- Epre_Ratio: enums.EpreRatio: Signaled 'epre-Ratio', PTRS relative to PDSCH.
- Resource_Offset: enums.ResourceOffset: Signaled 'resourceElementOffset'.

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DMRS:DL:PTRS
value: GetStruct = driver.configure.signaling.nradio.cell.dmrs.downlink.ptrs.
↪get(cell_name = 'abc')
```

Defines the IE 'PTRS-DownlinkConfig' for initial BWP.

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*structure: SetStruct*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DMRS:DL:PTRS
structure = driver.configure.signaling.nradio.cell.dmrs.downlink.ptrs.
↪SetStruct()
structure.Cell_Name: str = 'abc'
structure.Time_Dens_Pres: enums.DensityPreset = enums.DensityPreset.NPResent
structure.Time_Dens_Mcs_1: int = 1
structure.Time_Dens_Mcs_2: int = 1
structure.Time_Dens_Mcs_3: int = 1
structure.Freq_Dens_Pres: enums.DensityPreset = enums.DensityPreset.NPResent
structure.Freq_Dens_Nrb_0: int = 1
structure.Freq_Dens_Nrb_1: int = 1
structure.Epre_Ratio: enums.EpreRatio = enums.EpreRatio.R0
structure.Resource_Offset: enums.ResourceOffset = enums.ResourceOffset.NPResent
driver.configure.signaling.nradio.cell.dmrs.downlink.ptrs.set(structure)
```

Defines the IE 'PTRS-DownlinkConfig' for initial BWP.

param structure
for set value, see the help for SetStruct structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.dmrs.downlink.ptrs.clone()
```

Subgroups

6.3.4.10.2.340 Enable

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:DMRS:DL:PTRS:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DMRS:DL:PTRS:ENABle
value: bool = driver.configure.signaling.nradio.cell.dmrs.downlink.ptrs.enable.
↪get(cell_name = 'abc')
```

Enables sending the IE 'PTRS-DownlinkConfig' for initial BWP.

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DMRS:DL:PTRS:ENABle
driver.configure.signaling.nradio.cell.dmrs.downlink.ptrs.enable.set(cell_name␣
↪= 'abc', enable = False)
```

Enables sending the IE 'PTRS-DownlinkConfig' for initial BWP.

param cell_name
No help available

param enable
No help available

6.3.4.10.2.341 Uplink

class UplinkCls

Uplink commands group definition. 11 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.dmrs.uplink.clone()
```

Subgroups

6.3.4.10.2.342 Mta

class MtaCls

Mta commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.dmrs.uplink.mta.clone()
```

Subgroups

6.3.4.10.2.343 Length

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:DMRS:UL:MTA:LENGth
```

class LengthCls

Length commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → MaxLength

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DMRS:UL:MTA:LENGth
value: enums.MaxLength = driver.configure.signaling.nradio.cell.dmrs.uplink.mta.
↪length.get(cell_name = 'abc')
```

No command help available

param cell_name
No help available

return
max_length: No help available

set(*cell_name: str*, *max_length: MaxLength*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DMRS:UL:MTA:LENGth
driver.configure.signaling.nradio.cell.dmrs.uplink.mta.length.set(cell_name =
↪'abc', max_length = enums.MaxLength.L1)
```

No command help available

param cell_name
No help available

param max_length
No help available

6.3.4.10.2.344 Papr

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:DMRS:UL:MTA:PAPR
```

class PaprCls

Papr commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DMRS:UL:MTA:PAPR
value: bool = driver.configure.signaling.nradio.cell.dmrs.uplink.mta.papr.
↪get(cell_name = 'abc')
```

Enables the usage of a DMRS with a low PAPR, for PUSCH, mapping type A, initial BWP.

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DMRS:UL:MTA:PAPR
driver.configure.signaling.nradio.cell.dmrs.uplink.mta.papr.set(cell_name = 'abc
↪', enable = False)
```

Enables the usage of a DMRS with a low PAPR, for PUSCH, mapping type A, initial BWP.

param cell_name
No help available

param enable
No help available

6.3.4.10.2.345 Position

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:DMRS:UL:MTA:POSition
```

class PositionCls

Position commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → MtxPosition

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DMRS:UL:MTA:POSition
value: enums.MtxPosition = driver.configure.signaling.nradio.cell.dmrs.uplink.
↪mta.position.get(cell_name = 'abc')
```

Defines parameter 'dmrs-AdditionalPosition' for PUSCH, mapping type A, initial BWP.

param cell_name
No help available

return
position: No help available

set(*cell_name: str*, *position: MtxPosition*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DMRS:UL:MTA:POSition
driver.configure.signaling.nradio.cell.dmrs.uplink.mta.position.set(cell_name =
↪'abc', position = enums.MtxPosition.P0)
```

Defines parameter 'dmrs-AdditionalPosition' for PUSCH, mapping type A, initial BWP.

param cell_name
No help available

param position
No help available

6.3.4.10.2.346 TypePy

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:DMRS:UL:MTA:TYPE
```

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → ConfigType

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DMRS:UL:MTA:TYPE
value: enums.ConfigType = driver.configure.signaling.nradio.cell.dmrs.uplink.
↪mta.typePy.get(cell_name = 'abc')
```

No command help available

param cell_name
No help available

return
config_type: No help available

set(*cell_name: str*, *config_type: ConfigType*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DMRS:UL:MTA:TYPE
driver.configure.signaling.nradio.cell.dmrs.uplink.mta.typePy.set(cell_name =
↪'abc', config_type = enums.ConfigType.T1)
```

No command help available

param cell_name
No help available

param config_type
No help available

6.3.4.10.2.347 Mtb

class MtbCls

Mtb commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.dmrs.uplink.mtb.clone()
```

Subgroups

6.3.4.10.2.348 Length

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:DMRS:UL:MTB:LENGth
```

class LengthCls

Length commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → MaxLength

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DMRS:UL:MTB:LENGth
value: enums.MaxLength = driver.configure.signaling.nradio.cell.dmrs.uplink.mtb.
↪length.get(cell_name = 'abc')
```

No command help available

param cell_name
No help available

return
max_length: No help available

set(*cell_name: str*, *max_length: MaxLength*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DMRS:UL:MTB:LENGth
driver.configure.signaling.nradio.cell.dmrs.uplink.mtb.length.set(cell_name =
↪'abc', max_length = enums.MaxLength.L1)
```

No command help available

param cell_name
No help available

param max_length
No help available

6.3.4.10.2.349 Papr

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:DMRS:UL:MTB:PAPR
```

class PaprCls

Papr commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DMRS:UL:MTB:PAPR
value: bool = driver.configure.signaling.nradio.cell.dmrs.uplink.mtb.papr.
↪get(cell_name = 'abc')
```

Enables the usage of a DMRS with a low PAPR, for PUSCH, mapping type B, initial BWP.

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DMRS:UL:MTB:PAPR
driver.configure.signaling.nradio.cell.dmrs.uplink.mtb.papr.set(cell_name = 'abc
↪', enable = False)
```

Enables the usage of a DMRS with a low PAPR, for PUSCH, mapping type B, initial BWP.

param cell_name
No help available

param enable
No help available

6.3.4.10.2.350 Position

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:DMRS:UL:MTB:POSition
```

class PositionCls

Position commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → MtxPosition

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DMRS:UL:MTB:POSition
value: enums.MtxPosition = driver.configure.signaling.nradio.cell.dmrs.uplink.
↪mtb.position.get(cell_name = 'abc')
```

Defines parameter 'dmrs-AdditionalPosition' for PUSCH, mapping type B, initial BWP.

param cell_name
No help available

return
position: No help available

set(*cell_name: str*, *position: MtxPosition*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DMRS:UL:MTB:POSition
driver.configure.signaling.nradio.cell.dmrs.uplink.mtb.position.set(cell_name =
↪'abc', position = enums.MtxPosition.P0)
```

Defines parameter 'dmrs-AdditionalPosition' for PUSCH, mapping type B, initial BWP.

param cell_name
No help available

param position
No help available

6.3.4.10.2.351 TypePy

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:DMRS:UL:MTB:TYPE
```

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → ConfigType

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DMRS:UL:MTB:TYPE
value: enums.ConfigType = driver.configure.signaling.nradio.cell.dmrs.uplink.
↪mtb.typePy.get(cell_name = 'abc')
```

No command help available

param cell_name
No help available

return
config_type: No help available

set(*cell_name: str*, *config_type: ConfigType*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DMRS:UL:MTB:TYPE
driver.configure.signaling.nradio.cell.dmrs.uplink.mtb.typePy.set(cell_name =
↪'abc', config_type = enums.ConfigType.T1)
```

No command help available

param cell_name
No help available

param config_type
No help available

6.3.4.10.2.352 Ptrs

class PtrsCls

Ptrs commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.dmrs.uplink.ptrs.clone()
```

Subgroups

6.3.4.10.2.353 Enable

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:DMRS:UL:PTRS:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DMRS:UL:PTRS:ENABle
value: bool = driver.configure.signaling.nradio.cell.dmrs.uplink.ptrs.enable.
↪get(cell_name = 'abc')
```

Enables sending the IE 'PTRS-UplinkConfig' for initial BWP.

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DMRS:UL:PTRS:ENABle
driver.configure.signaling.nradio.cell.dmrs.uplink.ptrs.enable.set(cell_name =
↪'abc', enable = False)
```

Enables sending the IE 'PTRS-UplinkConfig' for initial BWP.

param cell_name
No help available

param enable
No help available

6.3.4.10.2.354 TpDisable

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:DMRS:UL:PTRS:TPDisable
```

class TpDisableCls

TpDisable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Time_Dens_Pres: enums.DensityPreset: Presence of the 'timeDensity' (field signaled or not) .
- Time_Dens_Mcs_1: int: Signaled 'timeDensity', ptrs-MCS1.
- Time_Dens_Mcs_2: int: Signaled 'timeDensity', ptrs-MCS2.
- Time_Dens_Mcs_3: int: Signaled 'timeDensity', ptrs-MCS3.
- Freq_Dens_Pres: enums.DensityPreset: Presence of the 'frequencyDensity' (field signaled or not) .
- Freq_Dens_Nrb_0: int: Signaled 'frequencyDensity', NRB0.
- Freq_Dens_Nrb_1: int: Signaled 'frequencyDensity', NRB1.
- Max_Ports: enums.MaxPorts: Signaled 'maxNrofPorts'.
- Resource_Offset: enums.ResourceOffset: Signaled 'resourceElementOffset'.
- Power: enums.PtrsPower: Signaled 'ptrs-Power'.

class SetStruct

Structure for setting input parameters. Fields:

- Cell_Name: str: No parameter help available
- Time_Dens_Pres: enums.DensityPreset: Presence of the 'timeDensity' (field signaled or not) .
- Time_Dens_Mcs_1: int: Signaled 'timeDensity', ptrs-MCS1.
- Time_Dens_Mcs_2: int: Signaled 'timeDensity', ptrs-MCS2.
- Time_Dens_Mcs_3: int: Signaled 'timeDensity', ptrs-MCS3.
- Freq_Dens_Pres: enums.DensityPreset: Presence of the 'frequencyDensity' (field signaled or not) .
- Freq_Dens_Nrb_0: int: Signaled 'frequencyDensity', NRB0.
- Freq_Dens_Nrb_1: int: Signaled 'frequencyDensity', NRB1.
- Max_Ports: enums.MaxPorts: Signaled 'maxNrofPorts'.
- Resource_Offset: enums.ResourceOffset: Signaled 'resourceElementOffset'.
- Power: enums.PtrsPower: Signaled 'ptrs-Power'.

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DMRS:UL:PTRS:TPDisable
value: GetStruct = driver.configure.signaling.nradio.cell.dmrs.uplink.ptrs.
↪tpDisable.get(cell_name = 'abc')
```

Defines the IE 'PTRS-UplinkConfig' for signals without transform precoding, initial BWP.

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*structure: SetStruct*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DMRS:UL:PTRS:TPDisable
structure = driver.configure.signaling.nradio.cell.dmrs.uplink.ptrs.tpDisable.
↪SetStruct()
structure.Cell_Name: str = 'abc'
structure.Time_Dens_Pres: enums.DensityPreset = enums.DensityPreset.NPResent
structure.Time_Dens_Mcs_1: int = 1
structure.Time_Dens_Mcs_2: int = 1
structure.Time_Dens_Mcs_3: int = 1
structure.Freq_Dens_Pres: enums.DensityPreset = enums.DensityPreset.NPResent
structure.Freq_Dens_Nrb_0: int = 1
structure.Freq_Dens_Nrb_1: int = 1
structure.Max_Ports: enums.MaxPorts = enums.MaxPorts.N1
structure.Resource_Offset: enums.ResourceOffset = enums.ResourceOffset.NPResent
structure.Power: enums.PtrsPower = enums.PtrsPower.P00
driver.configure.signaling.nradio.cell.dmrs.uplink.ptrs.tpDisable.set(structure)
```

Defines the IE 'PTRS-UplinkConfig' for signals without transform precoding, initial BWP.

param structure
for set value, see the help for SetStruct structure arguments.

6.3.4.10.2.355 TpEnable

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:DMRS:UL:PTRS:TPENable
```

class TpEnableCls

TpEnable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Sample_Dens_Nrb_0: int: Signaled 'sampleDensity', NRB0.
- Sample_Dens_Nrb_1: int: Signaled 'sampleDensity', NRB1.
- Sample_Dens_Nrb_2: int: Signaled 'sampleDensity', NRB2.
- Sample_Dens_Nrb_3: int: Signaled 'sampleDensity', NRB3.
- Sample_Dens_Nrb_4: int: Signaled 'sampleDensity', NRB4.
- Tp_Time_Dens: enums.TpTimeDens: Signaled 'timeDensityTransformPrecoding'.

class SetStruct

Structure for setting input parameters. Fields:

- Cell_Name: str: No parameter help available

- Sample_Dens_Nrb_0: int: Signaled 'sampleDensity', NRB0.
- Sample_Dens_Nrb_1: int: Signaled 'sampleDensity', NRB1.
- Sample_Dens_Nrb_2: int: Signaled 'sampleDensity', NRB2.
- Sample_Dens_Nrb_3: int: Signaled 'sampleDensity', NRB3.
- Sample_Dens_Nrb_4: int: Signaled 'sampleDensity', NRB4.
- Tp_Time_Dens: enums.TpTimeDens: Signaled 'timeDensityTransformPrecoding'.

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DMRS:UL:PTRS:TPENable
value: GetStruct = driver.configure.signaling.nradio.cell.dmrs.uplink.ptrs.
↪tpEnable.get(cell_name = 'abc')
```

Defines the IE 'PTRS-UplinkConfig' for signals with transform precoding, initial BWP.

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*structure: SetStruct*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DMRS:UL:PTRS:TPENable
structure = driver.configure.signaling.nradio.cell.dmrs.uplink.ptrs.tpEnable.
↪SetStruct()
structure.Cell_Name: str = 'abc'
structure.Sample_Dens_Nrb_0: int = 1
structure.Sample_Dens_Nrb_1: int = 1
structure.Sample_Dens_Nrb_2: int = 1
structure.Sample_Dens_Nrb_3: int = 1
structure.Sample_Dens_Nrb_4: int = 1
structure.Tp_Time_Dens: enums.TpTimeDens = enums.TpTimeDens.D2
driver.configure.signaling.nradio.cell.dmrs.uplink.ptrs.tpEnable.set(structure)
```

Defines the IE 'PTRS-UplinkConfig' for signals with transform precoding, initial BWP.

param structure
for set value, see the help for SetStruct structure arguments.

6.3.4.10.2.356 Downlink

class DownlinkCls

Downlink commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.downlink.clone()
```

Subgroups

6.3.4.10.2.357 LbWidth

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:DL:LBWidth
```

class LbWidthCls

LbWidth commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DL:LBWidth
value: int = driver.configure.signaling.nradio.cell.downlink.lbWidth.get(cell_
↪name = 'abc')
```

Defines the resource indication value (RIV) signaled as 'locationAndBandwidth', for the downlink, for the initial BWP.

param cell_name
No help available

return
riv: No help available

set(*cell_name: str*, *riv: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DL:LBWidth
driver.configure.signaling.nradio.cell.downlink.lbWidth.set(cell_name = 'abc',␣
↪riv = 1)
```

Defines the resource indication value (RIV) signaled as 'locationAndBandwidth', for the downlink, for the initial BWP.

param cell_name
No help available

param riv
No help available

6.3.4.10.2.358 Rb

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:DL:RB
```

class RbCls

Rb commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Number_Rb: int: No parameter help available
- Start_Rb: int: No parameter help available

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DL:RB
value: GetStruct = driver.configure.signaling.nradio.cell.downlink.rb.get(cell_
↪name = 'abc')
```

Defines the downlink of the initial BWP in the frequency domain as a contiguous set of RBs, within the carrier bandwidth.

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *number_rb: int*, *start_rb: int = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:DL:RB
driver.configure.signaling.nradio.cell.downlink.rb.set(cell_name = 'abc',␣
↪number_rb = 1, start_rb = 1)
```

Defines the downlink of the initial BWP in the frequency domain as a contiguous set of RBs, within the carrier bandwidth.

param cell_name
No help available

param number_rb
No help available

param start_rb
No help available

6.3.4.10.2.359 Harq

class HarqCls

Harq commands group definition. 18 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.harq.clone()
```

Subgroups

6.3.4.10.2.360 Downlink

class DownlinkCls

Downlink commands group definition. 9 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.harq.downlink.clone()
```

Subgroups

6.3.4.10.2.361 Auto

class AutoCls

Auto commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.harq.downlink.auto.clone()
```

Subgroups

6.3.4.10.2.362 Mret

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:HARQ:DL:AUTO:MRET
```

class MretCls

Mret commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:HARQ:DL:AUTO:MRET
value: int = driver.configure.signaling.nradio.cell.harq.downlink.auto.mret.
↪get(cell_name = 'abc')
```

Configures the maximum number of retransmissions, for auto-configured DL HARQ, for the initial BWP.

param cell_name
No help available

return
retransmissions: No help available

set(*cell_name: str*, *retransmissions: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:HARQ:DL:AUTO:MRET
driver.configure.signaling.nradio.cell.harq.downlink.auto.mret.set(cell_name =
↪'abc', retransmissions = 1)
```

Configures the maximum number of retransmissions, for auto-configured DL HARQ, for the initial BWP.

param cell_name
No help available

param retransmissions
No help available

6.3.4.10.2.363 Cmode

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:HARQ:DL:CMODe
```

class CmodeCls

Cmode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → ModeC

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:HARQ:DL:CMODe
value: enums.ModeC = driver.configure.signaling.nradio.cell.harq.downlink.cmode.
↪get(cell_name = 'abc')
```

Selects a mode for DL HARQ configuration, for the initial BWP.

param cell_name
No help available

return
mode: NOTC: no DL HARQ AUTO: automatic configuration of the DL HARQ settings USER: user-defined configuration of the DL HARQ settings

set(*cell_name: str*, *mode: ModeC*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:HARQ:DL:CMODe
driver.configure.signaling.nradio.cell.harq.downlink.cmode.set(cell_name = 'abc
↪', mode = enums.ModeC.AUTO)
```

Selects a mode for DL HARQ configuration, for the initial BWP.

param cell_name
No help available

param mode
NOTC: no DL HARQ AUTO: automatic configuration of the DL HARQ settings
USER: user-defined configuration of the DL HARQ settings

6.3.4.10.2.364 User

class UserCls

User commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.harq.downlink.user.clone()
```

Subgroups

6.3.4.10.2.365 Ack

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:HARQ:DL:USER:ACK
```

class AckCls

Ack commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → AckOrDtx

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:HARQ:DL:USER:ACK
value: enums.AckOrDtx = driver.configure.signaling.nradio.cell.harq.downlink.
↪user.ack.get(cell_name = 'abc')
```

Defines the reaction to ACKs sent by the UE, for user-defined DL HARQ, for the initial BWP.

param cell_name
No help available

return
ack: STOP: stop retransmitting CONTinue: continue retransmitting

set(*cell_name: str*, *ack: AckOrDtx*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:HARQ:DL:USER:ACK
driver.configure.signaling.nradio.cell.harq.downlink.user.ack.set(cell_name =
↪'abc', ack = enums.AckOrDtx.CONTinue)
```

Defines the reaction to ACKs sent by the UE, for user-defined DL HARQ, for the initial BWP.

param cell_name
No help available

param ack
STOP: stop retransmitting CONTinue: continue retransmitting

6.3.4.10.2.366 Dtx

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:HARQ:DL:USER:DTX
```

class DtxCls

Dtx commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → AckOrDtx

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:HARQ:DL:USER:DTX
value: enums.AckOrDtx = driver.configure.signaling.nradio.cell.harq.downlink.
↪user.dtx.get(cell_name = 'abc')
```

Defines the reaction to DTX (missing ACKs) , for user-defined DL HARQ, for the initial BWP.

param cell_name
No help available

return
dtx: STOP: stop retransmitting CONTinue: continue retransmitting

set(*cell_name: str*, *dtx: AckOrDtx*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:HARQ:DL:USER:DTX
driver.configure.signaling.nradio.cell.harq.downlink.user.dtx.set(cell_name =
↪'abc', dtx = enums.AckOrDtx.CONTinue)
```

Defines the reaction to DTX (missing ACKs) , for user-defined DL HARQ, for the initial BWP.

param cell_name
No help available

param dtx
STOP: stop retransmitting CONTinue: continue retransmitting

6.3.4.10.2.367 MinOffset

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:HARQ:DL:USER:MINoffset
```

class MinOffsetCls

MinOffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:HARQ:DL:USER:MINoffset
value: int = driver.configure.signaling.nradio.cell.harq.downlink.user.
↪minOffset.get(cell_name = 'abc')
```

Defines the minimum offset for user-defined DL HARQ, for the initial BWP.

param cell_name
No help available

return
offset: No help available

set(*cell_name: str*, *offset: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:HARQ:DL:USER:MINoffset
driver.configure.signaling.nradio.cell.harq.downlink.user.minOffset.set(cell_
↪name = 'abc', offset = 1)
```

Defines the minimum offset for user-defined DL HARQ, for the initial BWP.

param cell_name
No help available

param offset
No help available

6.3.4.10.2.368 Retransm

class RetransmCls

Retransm commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.harq.downlink.user.retransm.clone()
```

Subgroups

6.3.4.10.2.369 Ariv

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:HARQ:DL:USER:RETRansm:ARIV
```

class ArivCls

Ariv commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *index: int*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:HARQ:DL:USER:RETRansm:ARIV
value: bool = driver.configure.signaling.nradio.cell.harq.downlink.user.
↪retransm.ariv.get(cell_name = 'abc', index = 1)
```

Configures auto RIV for a certain retransmission, for user-defined DL HARQ, for the initial BWP.

param cell_name
No help available

param index
Index of the retransmission

return
riv: ON: Auto RIV enabled, no. of RB and start RB set automatically. OFF: Auto RIV disabled, you can define no. of RB and start RB.

set(*cell_name: str*, *index: int*, *riv: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:HARQ:DL:USER:RETRansm:ARIV
driver.configure.signaling.nradio.cell.harq.downlink.user.retransm.ariv.
↪set(cell_name = 'abc', index = 1, riv = False)
```

Configures auto RIV for a certain retransmission, for user-defined DL HARQ, for the initial BWP.

param cell_name
No help available

param index
Index of the retransmission

param riv
ON: Auto RIV enabled, no. of RB and start RB set automatically. OFF: Auto RIV disabled, you can define no. of RB and start RB.

6.3.4.10.2.370 Modulation

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:HARQ:DL:USER:RETRansm:MODulation
```

class ModulationCls

Modulation commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *index: int*) → ModulationRetr

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:HARQ:DL:USER:RETRansm:MODulation
value: enums.ModulationRetr = driver.configure.signaling.nradio.cell.harq.
↪downlink.user.retransm.modulation.get(cell_name = 'abc', index = 1)
```

Selects a modulation scheme for a certain retransmission, for user-defined DL HARQ, for the initial BWP.

param cell_name
No help available

param index
Index of the retransmission

return
modulation: BPSK, auto mode, /2-BPSK, QPSK, 16QAM, 64QAM, 256QAM, 1024QAM

set(*cell_name: str*, *index: int*, *modulation: ModulationRetr*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:HARQ:DL:USER:RETRansm:MODulation
driver.configure.signaling.nradio.cell.harq.downlink.user.retransm.modulation.
↪set(cell_name = 'abc', index = 1, modulation = enums.ModulationRetr.AUTO)
```

Selects a modulation scheme for a certain retransmission, for user-defined DL HARQ, for the initial BWP.

param cell_name
No help available

param index
Index of the retransmission

param modulation
BPSK, auto mode, /2-BPSK, QPSK, 16QAM, 64QAM, 256QAM, 1024QAM

6.3.4.10.2.371 Rb

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:HARQ:DL:USER:RETRansm:RB
```

class RbCls

Rb commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Nrb: int: No parameter help available
- Start_Rb: int: No parameter help available

get(*cell_name: str*, *index: int*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:HARQ:DL:USER:RETRansm:RB
value: GetStruct = driver.configure.signaling.nradio.cell.harq.downlink.user.
↪retransm.rb.get(cell_name = 'abc', index = 1)
```

Configures the number of RB and start RB for a certain retransmission, for user-defined DL HARQ, for the initial BWP. Only relevant for disabled auto RIV.

param cell_name
No help available

param index
Index of the retransmission

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *index: int*, *nrb: int*, *start_rb: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:HARQ:DL:USER:RETRansm:RB
driver.configure.signaling.nradio.cell.harq.downlink.user.retransm.rb.set(cell_
↪name = 'abc', index = 1, nrb = 1, start_rb = 1)
```

Configures the number of RB and start RB for a certain retransmission, for user-defined DL HARQ, for the initial BWP. Only relevant for disabled auto RIV.

param cell_name
No help available

param index
Index of the retransmission

param nrb
No help available

param start_rb
No help available

6.3.4.10.2.372 Rversion

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:HARQ:DL:USER:RETRansm:RVERsion
```

class RversionCls

Rversion commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *index: int*) → Version

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:HARQ:DL:USER:RETRansm:RVERsion
value: enums.Version = driver.configure.signaling.nradio.cell.harq.downlink.
↪user.retransm.rversion.get(cell_name = 'abc', index = 1)
```

Selects a redundancy version for a certain retransmission, for user-defined DL HARQ, for the initial BWP.

param cell_name
No help available

param index
Index of the retransmission

return
version: Auto mode, redundancy version number 0 to 3.

set(*cell_name: str*, *index: int*, *version: Version*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:HARQ:DL:USER:RETRansm:RVERsion
driver.configure.signaling.nradio.cell.harq.downlink.user.retransm.rversion.
↪set(cell_name = 'abc', index = 1, version = enums.Version.AUTO)
```

Selects a redundancy version for a certain retransmission, for user-defined DL HARQ, for the initial BWP.

param cell_name
No help available

param index
Index of the retransmission

param version
Auto mode, redundancy version number 0 to 3.

6.3.4.10.2.373 Uplink

class UplinkCls

Uplink commands group definition. 9 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.harq.uplink.clone()
```

Subgroups

6.3.4.10.2.374 Auto

class AutoCls

Auto commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.harq.uplink.auto.clone()
```

Subgroups

6.3.4.10.2.375 Mret

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:HARQ:UL:AUTO:MRET
```

class MretCls

Mret commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:HARQ:UL:AUTO:MRET
value: int = driver.configure.signaling.nradio.cell.harq.uplink.auto.mret.
↪get(cell_name = 'abc')
```

Configures the maximum number of retransmissions, for auto-configured UL HARQ, for the initial BWP.

param cell_name
No help available

return
retransmissions: No help available

set(*cell_name: str*, *retransmissions: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:HARQ:UL:AUTO:MRET
driver.configure.signaling.nradio.cell.harq.uplink.auto.mret.set(cell_name =
↪'abc', retransmissions = 1)
```

Configures the maximum number of retransmissions, for auto-configured UL HARQ, for the initial BWP.

param cell_name
No help available

param retransmissions
No help available

6.3.4.10.2.376 Behavior

class BehaviorCls

Behavior commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.harq.uplink.behavior.clone()
```

Subgroups

6.3.4.10.2.377 CrcPass

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:HARQ:UL:BEHavior:CRCPass
```

class CrcPassCls

CrcPass commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → AckOrDtx

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:HARQ:UL:BEHavior:CRCPass
value: enums.AckOrDtx = driver.configure.signaling.nradio.cell.harq.uplink.
↪behavior.crcPass.get(cell_name = 'abc')
```

Defines the behavior when a UL transmission passes the CRC check: stop or continue requesting retransmissions from the UE, for the initial BWP.

param cell_name
No help available

return
behavior: No help available

set(*cell_name: str*, *behavior: AckOrDtx*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:HARQ:UL:BEHavior:CRCPass
driver.configure.signaling.nradio.cell.harq.uplink.behavior.crcPass.set(cell_
↪name = 'abc', behavior = enums.AckOrDtx.CONTinue)
```

Defines the behavior when a UL transmission passes the CRC check: stop or continue requesting retransmissions from the UE, for the initial BWP.

param cell_name
No help available

param behavior
No help available

6.3.4.10.2.378 NulPower

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:HARQ:UL:BEHavior:NULPower
```

class NulPowerCls

NulPower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → AckOrDtx

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:HARQ:UL:BEHavior:NULPower
value: enums.AckOrDtx = driver.configure.signaling.nradio.cell.harq.uplink.
↪behavior.nulPower.get(cell_name = 'abc')
```

Defines the behavior when detecting no UL power: stop or continue requesting retransmissions from the UE, for the initial BWP.

param cell_name
No help available

return
behavior: No help available

set(*cell_name: str*, *behavior: AckOrDtx*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:HARQ:UL:BEHavior:NULPower
driver.configure.signaling.nradio.cell.harq.uplink.behavior.nulPower.set(cell_
↪name = 'abc', behavior = enums.AckOrDtx.CONTinue)
```

Defines the behavior when detecting no UL power: stop or continue requesting retransmissions from the UE, for the initial BWP.

param cell_name
No help available

param behavior
No help available

6.3.4.10.2.379 Cmode

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:HARQ:UL:CMODe
```

class CmodeCls

Cmode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → ModeC

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:HARQ:UL:CMODe
value: enums.ModeC = driver.configure.signaling.nradio.cell.harq.uplink.cmode.
↪get(cell_name = 'abc')
```

Selects a mode for UL HARQ configuration, for the initial BWP.

param cell_name
No help available

return
mode: NOTC: no UL HARQ AUTO: automatic configuration of the UL HARQ settings USER: user-defined configuration of the UL HARQ settings

set(*cell_name: str*, *mode: ModeC*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:HARQ:UL:CMODe
driver.configure.signaling.nradio.cell.harq.uplink.cmode.set(cell_name = 'abc',␣
↪mode = enums.ModeC.AUTO)
```

Selects a mode for UL HARQ configuration, for the initial BWP.

param cell_name
No help available

param mode
NOTC: no UL HARQ AUTO: automatic configuration of the UL HARQ settings USER: user-defined configuration of the UL HARQ settings

6.3.4.10.2.380 User

class UserCls

User commands group definition. 5 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.harq.uplink.user.clone()
```

Subgroups

6.3.4.10.2.381 Retransm

class RetransmCls

Retransm commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.harq.uplink.user.retransm.clone()
```

Subgroups

6.3.4.10.2.382 Ariv

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:HARQ:UL:USER:RETRansm:ARIV
```

class ArivCls

Ariv commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *index: int*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:HARQ:UL:USER:RETRansm:ARIV
value: bool = driver.configure.signaling.nradio.cell.harq.uplink.user.retransm.
↪ariv.get(cell_name = 'abc', index = 1)
```

Configures auto RIV for a certain retransmission, for user-defined UL HARQ, for the initial BWP.

param cell_name
No help available

param index
Index of the retransmission

return
riv: ON: Auto RIV enabled, no. of RB and start RB set automatically. OFF: Auto RIV disabled, you can define no. of RB and start RB.

set(*cell_name: str*, *index: int*, *riv: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:HARQ:UL:USER:RETRansm:ARIV
driver.configure.signaling.nradio.cell.harq.uplink.user.retransm.ariv.set(cell_
↪name = 'abc', index = 1, riv = False)
```

Configures auto RIV for a certain retransmission, for user-defined UL HARQ, for the initial BWP.

param cell_name
No help available

param index
Index of the retransmission

param riv
ON: Auto RIV enabled, no. of RB and start RB set automatically. OFF: Auto RIV disabled, you can define no. of RB and start RB.

6.3.4.10.2.383 Modulation

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:HARQ:UL:USER:RETRansm:MODulation
```

class ModulationCls

Modulation commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *index: int*) → ModulationRetr

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:HARQ:UL:USER:RETRansm:MODulation
value: enums.ModulationRetr = driver.configure.signaling.nradio.cell.harq.
↪uplink.user.retransm.modulation.get(cell_name = 'abc', index = 1)
```

Selects a modulation scheme for a certain retransmission, for user-defined UL HARQ, for the initial BWP.

param cell_name
No help available

param index
Index of the retransmission

return
modulation: BPSK, auto mode, /2-BPSK, QPSK, 16QAM, 64QAM, 256QAM, 1024QAM

set(*cell_name: str*, *index: int*, *modulation: ModulationRetr*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:HARQ:UL:USER:RETRansm:MODulation
driver.configure.signaling.nradio.cell.harq.uplink.user.retransm.modulation.
↪set(cell_name = 'abc', index = 1, modulation = enums.ModulationRetr.AUTO)
```

Selects a modulation scheme for a certain retransmission, for user-defined UL HARQ, for the initial BWP.

param cell_name
No help available

param index
Index of the retransmission

param modulation
BPSK, auto mode, /2-BPSK, QPSK, 16QAM, 64QAM, 256QAM, 1024QAM

6.3.4.10.2.384 Moffset

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:HARQ:UL:USER:RETRansm:MOFFset
```

class MoffsetCls

Moffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *index: int*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:HARQ:UL:USER:RETRansm:MOFFset
value: int = driver.configure.signaling.nradio.cell.harq.uplink.user.retransm.
↪moffset.get(cell_name = 'abc', index = 1)
```

Minimum number of slots between feedback processing and sending the retransmission DCI, for the initial BWP.

param cell_name
No help available

param index
Index of the retransmission

return
minimum_offset: No help available

set(*cell_name: str*, *index: int*, *minimum_offset: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:HARQ:UL:USER:RETRansm:MOFFset
driver.configure.signaling.nradio.cell.harq.uplink.user.retransm.moffset.
↪set(cell_name = 'abc', index = 1, minimum_offset = 1)
```

Minimum number of slots between feedback processing and sending the retransmission DCI, for the initial BWP.

param cell_name
No help available

param index
Index of the retransmission

param minimum_offset
No help available

6.3.4.10.2.385 Rb

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:HARQ:UL:USER:RETRansm:RB
```

class RbCls

Rb commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Nrb: int: No parameter help available
- Start_Rb: int: No parameter help available

get(*cell_name: str*, *index: int*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:HARQ:UL:USER:RETRansm:RB
value: GetStruct = driver.configure.signaling.nradio.cell.harq.uplink.user.
↪retransm.rb.get(cell_name = 'abc', index = 1)
```

Configures the number of RB and start RB for a certain retransmission, for user-defined UL HARQ, for the initial BWP. Only relevant for disabled auto RIV.

param cell_name
No help available

param index
Index of the retransmission

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *index: int*, *nrb: int*, *start_rb: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:HARQ:UL:USER:RETRansm:RB
driver.configure.signaling.nradio.cell.harq.uplink.user.retransm.rb.set(cell_
↪name = 'abc', index = 1, nrb = 1, start_rb = 1)
```

Configures the number of RB and start RB for a certain retransmission, for user-defined UL HARQ, for the initial BWP. Only relevant for disabled auto RIV.

param cell_name
No help available

param index
Index of the retransmission

param nrb
No help available

param start_rb
No help available

6.3.4.10.2.386 Rversion

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:HARQ:UL:USER:RETRansm:RVERsion
```

class RversionCls

Rversion commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *index: int*) → Version

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:HARQ:UL:USER:RETRansm:RVERsion
value: enums.Version = driver.configure.signaling.nradio.cell.harq.uplink.user.
↪retransm.rversion.get(cell_name = 'abc', index = 1)
```

Selects a redundancy version for a certain retransmission, for user-defined UL HARQ, for the initial BWP.

param cell_name
No help available

param index
Index of the retransmission

return
version: Auto mode, redundancy version number 0 to 3.

set(*cell_name: str*, *index: int*, *version: Version*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:HARQ:UL:USER:RETRansm:RVERsion
driver.configure.signaling.nradio.cell.harq.uplink.user.retransm.rversion.
↪set(cell_name = 'abc', index = 1, version = enums.Version.AUTO)
```

Selects a redundancy version for a certain retransmission, for user-defined UL HARQ, for the initial BWP.

param cell_name
No help available

param index
Index of the retransmission

param version
Auto mode, redundancy version number 0 to 3.

6.3.4.10.2.387 Ibwp

class IbwpCls

Ibwp commands group definition. 5 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.ibwp.clone()
```

Subgroups

6.3.4.10.2.388 Coreset

class CoresetCls

Coreset commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.ibwp.coreset.clone()
```

Subgroups

6.3.4.10.2.389 Duration

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:IBWP:COReset:DURation
```

class DurationCls

Duration commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → Spreset

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:IBWP:COReset:DURation
value: enums.Spreset = driver.configure.signaling.nradio.cell.ibwp.coreset.
↪duration.get(cell_name = 'abc')
```

Specifies the duration of the CORESET 1, in PDCCH symbols.

param cell_name
No help available

return
duration: No help available

set(*cell_name: str*, *duration: Spreset*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:IBWP:COReset:DURation
driver.configure.signaling.nradio.cell.ibwp.coreset.duration.set(cell_name =
↪'abc', duration = enums.Spreset.S1)
```

Specifies the duration of the CORESET 1, in PDCCH symbols.

param cell_name
No help available

param duration
No help available

6.3.4.10.2.390 FdrBitmap

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:IBWP:COReset:FDRBitmap
```

class FdrBitmapCls

FdrBitmap commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → str

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:IBWP:COReset:FDRBitmap
value: str = driver.configure.signaling.nradio.cell.ibwp.coreset.fdrBitmap.
↪get(cell_name = 'abc')
```

Specifies the frequency domain resources for the CORESET 1, as a bitmap with 45 bits.

param cell_name
No help available

return
bitmap: No help available

set(*cell_name: str*, *bitmap: str*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:IBWP:COReset:FDRBitmap
driver.configure.signaling.nradio.cell.ibwp.coreset.fdrBitmap.set(cell_name =
↪'abc', bitmap = 'abc')
```

Specifies the frequency domain resources for the CORESET 1, as a bitmap with 45 bits.

param cell_name
No help available

param bitmap
No help available

6.3.4.10.2.391 Ncandidates

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:IBWP:COReset:NCANdidates
```

class NcandidatesCls

Ncandidates commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Aggr_Level_1: enums.AggrLevel: No parameter help available
- Aggr_Level_2: enums.AggrLevel: No parameter help available
- Aggr_Level_4: enums.AggrLevel: No parameter help available
- Aggr_Level_8: enums.AggrLevel: No parameter help available
- Aggr_Level_16: enums.AggrLevel: No parameter help available

class SetStruct

Structure for setting input parameters. Contains optional setting parameters. Fields:

- Cell_Name: str: No parameter help available
- Aggr_Level_1: enums.AggrLevel: No parameter help available
- Aggr_Level_2: enums.AggrLevel: No parameter help available
- Aggr_Level_4: enums.AggrLevel: No parameter help available

- Aggr_Level_8: enums.AggrLevel: No parameter help available
- Aggr_Level_16: enums.AggrLevel: No parameter help available

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:IBWP:COReset:NCANdidates
value: GetStruct = driver.configure.signaling.nradio.cell.ibwp.coreset.
↪ncandidates.get(cell_name = 'abc')
```

Configures the number of PDCCH candidates per aggregation level.

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*structure: SetStruct*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:IBWP:COReset:NCANdidates
structure = driver.configure.signaling.nradio.cell.ibwp.coreset.ncandidates.
↪SetStruct()
structure.Cell_Name: str = 'abc'
structure.Aggr_Level_1: enums.AggrLevel = enums.AggrLevel.N0
structure.Aggr_Level_2: enums.AggrLevel = enums.AggrLevel.N0
structure.Aggr_Level_4: enums.AggrLevel = enums.AggrLevel.N0
structure.Aggr_Level_8: enums.AggrLevel = enums.AggrLevel.N0
structure.Aggr_Level_16: enums.AggrLevel = enums.AggrLevel.N0
driver.configure.signaling.nradio.cell.ibwp.coreset.ncandidates.set(structure)
```

Configures the number of PDCCH candidates per aggregation level.

param structure
for set value, see the help for SetStruct structure arguments.

6.3.4.10.2.392 Rmatching

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:IBWP:COReset:RMATching
```

class RmatchingCls

Rmatching commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:IBWP:COReset:RMATching
value: bool = driver.configure.signaling.nradio.cell.ibwp.coreset.rmatching.
↪get(cell_name = 'abc')
```

Enables or disables the usage of the CORESET 1 as a PDSCH rate-matching pattern.

param cell_name
No help available

return
pdsch_enable: No help available

set(*cell_name: str*, *pdsch_enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:IBWP:COReset:RMATching
driver.configure.signaling.nradio.cell.ibwp.coreset.rmatching.set(cell_name =
↪'abc', pdsch_enable = False)
```

Enables or disables the usage of the CORESET 1 as a PDSCH rate-matching pattern.

param cell_name
No help available

param pdsch_enable
No help available

6.3.4.10.2.393 Rcap

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:IBWP:RCAP
```

class RcapCls

Rcap commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:IBWP:RCAP
value: bool = driver.configure.signaling.nradio.cell.ibwp.rcap.get(cell_name =
↪'abc')
```

Selects whether the initial BWP is specific for RedCap or not.

param cell_name
No help available

return
enable: - ON: The initial BWP is specific for RedCap. It is signaled via 'initialUplinkBWP-RedCap-r17' and 'initialDownlinkBWP-RedCap-r17'. - OFF: The initial BWP is not specific for RedCap. It is signaled via 'initialUplinkBWP' and 'initialDownlinkBWP'.

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:IBWP:RCAP
driver.configure.signaling.nradio.cell.ibwp.rcap.set(cell_name = 'abc', enable␣
↪= False)
```

Selects whether the initial BWP is specific for RedCap or not.

param cell_name
No help available

param enable

- ON: The initial BWP is specific for RedCap. It is signaled via 'initialUplinkBWP-RedCap-r17' and 'initialDownlinkBWP-RedCap-r17'.
- OFF: The initial BWP is not specific for RedCap. It is signaled via 'initialUplinkBWP' and 'initialDownlinkBWP'.

6.3.4.10.2.394 Info

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:INFO
```

class InfoCls

Info commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Phys_Cell_Id: int: No parameter help available
- Name_Ta: List[str]: No parameter help available

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:INFO
value: GetStruct = driver.configure.signaling.nradio.cell.info.get(cell_name =
↪'abc')
```

No command help available

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

6.3.4.10.2.395 Mconfig

class MconfigCls

Mconfig commands group definition. 7 total commands, 7 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.mconfig.clone()
```

Subgroups

6.3.4.10.2.396 Aoa

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:MCONfig:AOA
```

class AoaCls

Aoa commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:MCONfig:AOA
value: int = driver.configure.signaling.nradio.cell.mconfig.aoa.get(cell_name =
↪'abc')
```

Selects the maximum number of emulated angles of arrival.

param cell_name
No help available

return
aoa: No help available

set(*cell_name: str*, *aoa: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:MCONfig:AOA
driver.configure.signaling.nradio.cell.mconfig.aoa.set(cell_name = 'abc', aoa =␣
↪1)
```

Selects the maximum number of emulated angles of arrival.

param cell_name
No help available

param aoa
No help available

6.3.4.10.2.397 Aports

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:MCONfig:APORts
```

class AportsCls

Aports commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:MCONfig:APORts
value: int = driver.configure.signaling.nradio.cell.mconfig.aports.get(cell_
↪name = 'abc')
```

Selects the maximum number of UL antenna ports at the instrument side, allowed in live mode.

param cell_name
No help available

return
ul_antenna_ports: No help available

set(*cell_name: str*, *ul_antenna_ports: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:MCONfig:APORts
driver.configure.signaling.nradio.cell.mconfig.aports.set(cell_name = 'abc', ul_
↪antenna_ports = 1)
```

Selects the maximum number of UL antenna ports at the instrument side, allowed in live mode.

param cell_name
No help available

param ul_antenna_ports
No help available

6.3.4.10.2.398 Bandwidth

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:MCONfig:BWIDth
```

class BandwidthCls

Bandwidth commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → DlUlBandwidth

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:MCONfig:BWIDth
value: enums.DlUlBandwidth = driver.configure.signaling.nradio.cell.mconfig.
↪bandwidth.get(cell_name = 'abc')
```

Selects the maximum carrier bandwidth allowed in live mode.

param cell_name
No help available

return
bandwidth: Bandwidth in MHz

set(*cell_name: str*, *bandwidth: DlUlBandwidth*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:MCONfig:BWIDth
driver.configure.signaling.nradio.cell.mconfig.bandwidth.set(cell_name = 'abc',␣
↪bandwidth = enums.DlUlBandwidth.B005)
```

Selects the maximum carrier bandwidth allowed in live mode.

param cell_name
No help available

param bandwidth
Bandwidth in MHz

6.3.4.10.2.399 Cdeployment

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:MCONfig:CDEPloyment
```

class CdeploymentCls

Cdeployment commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → CellDeployment

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:MCONfig:CDEPloyment
value: enums.CellDeployment = driver.configure.signaling.nradio.cell.mconfig.
↪cdeployment.get(cell_name = 'abc')
```

Selects whether the cell is a real cell or a virtual cell.

param cell_name
No help available

return
cell_deployment: No help available

set(*cell_name: str*, *cell_deployment: CellDeployment*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:MCONfig:CDEPloyment
driver.configure.signaling.nradio.cell.mconfig.cdeployment.set(cell_name = 'abc
↪', cell_deployment = enums.CellDeployment.REAL)
```

Selects whether the cell is a real cell or a virtual cell.

param cell_name
No help available

param cell_deployment
No help available

6.3.4.10.2.400 CsirsPorts

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:MCONfig:CSIRsports
```

class CsirsPortsCls

CsirsPorts commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → Ports

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:MCONfig:CSIRsports
value: enums.Ports = driver.configure.signaling.nradio.cell.mconfig.csirsPorts.
↪get(cell_name = 'abc')
```

Selects the maximum number of CSI-RS antenna ports allowed in live mode.

param cell_name
No help available

return
ant_no_ports: No help available

set(*cell_name: str*, *ant_no_ports: Ports*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:MCONfig:CSIRsports
driver.configure.signaling.nradio.cell.mconfig.csirsPorts.set(cell_name = 'abc',
↪ ant_no_ports = enums.Ports.P1)
```

Selects the maximum number of CSI-RS antenna ports allowed in live mode.

param cell_name
No help available

param ant_no_ports
No help available

6.3.4.10.2.401 Modulation

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:MCONfig:MODulation
```

class ModulationCls

Modulation commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → Modulation

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:MCONfig:MODulation
value: enums.Modulation = driver.configure.signaling.nradio.cell.mconfig.
↪modulation.get(cell_name = 'abc')
```

Selects the maximum UL modulation scheme allowed in live mode.

param cell_name
No help available

return
modulation: No help available

set(*cell_name: str*, *modulation: Modulation*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:MCONfig:MODulation
driver.configure.signaling.nradio.cell.mconfig.modulation.set(cell_name = 'abc',
↪ modulation = enums.Modulation.BPSK)
```

Selects the maximum UL modulation scheme allowed in live mode.

param cell_name
No help available

param modulation
No help available

6.3.4.10.2.402 Sspacing

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:MCONfig:SSPacing
```

class SspacingCls

Sspacing commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:MCONfig:SSPacing
value: int = driver.configure.signaling.nradio.cell.mconfig.sspacing.get(cell_
↪name = 'abc')
```

Selects the maximum subcarrier spacing allowed in live mode.

param cell_name
No help available

return
spacing: No help available

set(*cell_name: str*, *spacing: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:MCONfig:SSPacing
driver.configure.signaling.nradio.cell.mconfig.sspacing.set(cell_name = 'abc',␣
↪spacing = 1)
```

Selects the maximum subcarrier spacing allowed in live mode.

param cell_name
No help available

param spacing
No help available

6.3.4.10.2.403 Msg

class MsgCls

Msg commands group definition. 3 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.msg.clone()
```

Subgroups

6.3.4.10.2.404 Tdomain

class TdomainCls

Tdomain commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.msg.tdomain.clone()
```

Subgroups

6.3.4.10.2.405 Chmapping

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:MSG<id>:TDOMain:CHMapping
```

class ChmappingCls

Chmapping commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → Mapping

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:MSG<id>:TDOMain:CHMapping
value: enums.Mapping = driver.configure.signaling.nradio.cell.msg.tdomain.
↪chmapping.get(cell_name = 'abc')
```

Configures the PUSCH mapping type A or B for msg3.

param cell_name
No help available

return
mapping: No help available

set(*cell_name: str*, *mapping: Mapping*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:MSG<id>:TDOMain:CHMapping
driver.configure.signaling.nradio.cell.msg.tdomain.chmapping.set(cell_name =
↪'abc', mapping = enums.Mapping.A)
```

Configures the PUSCH mapping type A or B for msg3.

param cell_name
No help available

param mapping
No help available

6.3.4.10.2.406 Ktwo

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:MSG<id>:TDOMain:KTWO
```

class KtwoCls

Ktwo commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:MSG<id>:TDOMain:KTWO
value: int = driver.configure.signaling.nradio.cell.msg.tdomain.ktwo.get(cell_
↪name = 'abc')
```

Configures k2 influencing the slot offset between msg2 and msg3.

param cell_name
No help available

return
k_2: No help available

set(*cell_name: str*, *k_2: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:MSG<id>:TDOMain:KTWO
driver.configure.signaling.nradio.cell.msg.tdomain.ktwo.set(cell_name = 'abc',␣
↪k_2 = 1)
```

Configures k2 influencing the slot offset between msg2 and msg3.

param cell_name
No help available

param k_2
No help available

6.3.4.10.2.407 Symbol

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:MSG<id>:TDOMain:SYMBol
```

class SymbolCls

Symbol commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Number_Symbol: int: Number of allocated OFDM symbols.
- Start_Symbol: int: Index of the first allocated OFDM symbol.

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:MSG<id>:TDOMain:SYMBol
value: GetStruct = driver.configure.signaling.nradio.cell.msg.tdomain.symbol.
↪get(cell_name = 'abc')
```

Configures the allocated OFDM symbols for msg3.

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *number_symbol: int*, *start_symbol: int = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:MSG<id>:TDOMain:SYMBol
driver.configure.signaling.nradio.cell.msg.tdomain.symbol.set(cell_name = 'abc',
↪ number_symbol = 1, start_symbol = 1)
```

Configures the allocated OFDM symbols for msg3.

param cell_name
No help available

param number_symbol
Number of allocated OFDM symbols.

param start_symbol
Index of the first allocated OFDM symbol.

6.3.4.10.2.408 Nssb

class NssbCls

Nssb commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.nssb.clone()
```

Subgroups

6.3.4.10.2.409 Arfcn

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:NSSB:ARFCn
```

class ArfcnCls

Arfcn commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:NSSB:ARFCn
value: int = driver.configure.signaling.nradio.cell.nssb.arfcn.get(cell_name =
↪'abc')
```

Configures the channel number of the NCD-SSB signaled as 'absoluteFrequencySSB-r17', for the initial BWP.

param cell_name
No help available

return
number: No help available

set(*cell_name: str*, *number: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:NSSB:ARFCn
driver.configure.signaling.nradio.cell.nssb.arfcn.set(cell_name = 'abc', number␣
↪= 1)
```

Configures the channel number of the NCD-SSB signaled as 'absoluteFrequencySSB-r17', for the initial BWP.

param cell_name
No help available

param number
No help available

6.3.4.10.2.410 Enable

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:NSSB:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:NSSB:ENABle
value: bool = driver.configure.signaling.nradio.cell.nssb.enable.get(cell_name␣
↪= 'abc')
```

Enables sending the IE 'NonCellDefiningSSB-r17', for the initial BWP.

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:NSSB:ENABle
driver.configure.signaling.nradio.cell.nssb.enable.set(cell_name = 'abc',␣
↪enable = False)
```

Enables sending the IE 'NonCellDefiningSSB-r17', for the initial BWP.

param cell_name
No help available

param enable
No help available

6.3.4.10.2.411 Offset

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:NSSB:OFFSet
```

class OffsetCls

Offset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → TimeOffset

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:NSSB:OFFSet
value: enums.TimeOffset = driver.configure.signaling.nradio.cell.nssb.offset.
↪get(cell_name = 'abc')
```

Configures the time offset between the first burst of the NCD-SSB and the first burst of the CD-SSB, signaled as 'ssb-TimeOffset-r17', for the initial BWP.

param cell_name
No help available

return
time_offset: No help available

set(*cell_name: str*, *time_offset: TimeOffset*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:NSSB:OFFSet
driver.configure.signaling.nradio.cell.nssb.offset.set(cell_name = 'abc', time_
↪offset = enums.TimeOffset.T0)
```

Configures the time offset between the first burst of the NCD-SSB and the first burst of the CD-SSB, signaled as 'ssb-TimeOffset-r17', for the initial BWP.

param cell_name
No help available

param time_offset
No help available

6.3.4.10.2.412 Periodicity

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:NSSB:PERiodicity
```

class PeriodicityCls

Periodicity commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → PeriodicityB

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:NSSB:PERiodicity
value: enums.PeriodicityB = driver.configure.signaling.nradio.cell.nssb.
↪periodicity.get(cell_name = 'abc')
```

Configures the periodicity of the NCD-SSB signaled as 'ssb-Periodicity-r17', for the initial BWP.

param cell_name
No help available

return
periodicity: No help available

set(*cell_name: str*, *periodicity: PeriodicityB*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:NSSB:PERiodicity
driver.configure.signaling.nradio.cell.nssb.periodicity.set(cell_name = 'abc',␣
↪periodicity = enums.PeriodicityB.P10)
```

Configures the periodicity of the NCD-SSB signaled as 'ssb-Periodicity-r17', for the initial BWP.

param cell_name
No help available

param periodicity
No help available

6.3.4.10.2.413 Pcid

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:PCID
```

class PcidCls

Pcid commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:PCID
value: int = driver.configure.signaling.nradio.cell.pcid.get(cell_name = 'abc')
```

Defines the physical cell ID.

param cell_name
No help available

return
idn: No help available

set(*cell_name: str*, *idn: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:PCID
driver.configure.signaling.nradio.cell.pcid.set(cell_name = 'abc', idn = 1)
```

Defines the physical cell ID.

param cell_name
No help available

param idn
No help available

6.3.4.10.2.414 Pcycle

class PcycleCls

Pcycle commands group definition. 5 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.pcycle.clone()
```

Subgroups

6.3.4.10.2.415 EdRx

class EdRxCls

EdRx commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.pcycle.edRx.clone()
```

Subgroups

6.3.4.10.2.416 Aidle

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:PCYCle:EDRX:AIDLe
```

class AidleCls

Aidle commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:PCYCle:EDRX:AIDLe
value: bool = driver.configure.signaling.nradio.cell.pcycle.edRx.aidle.get(cell_
↪name = 'abc')
```

Configures 'eDRX-AllowedIdle' in SIB1.

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:PCYCle:EDRX:AIDLe
driver.configure.signaling.nradio.cell.pcycle.edRx.aidle.set(cell_name = 'abc',␣
↪enable = False)
```

Configures 'eDRX-AllowedIdle' in SIB1.

param cell_name
No help available

param enable
No help available

6.3.4.10.2.417 Ainactive

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:PCYCle:EDRX:AINactive
```

class AinactiveCls

Ainactive commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:PCYCle:EDRX:AINactive
value: bool = driver.configure.signaling.nradio.cell.pcycle.edRx.ainactive.
↪get(cell_name = 'abc')
```

Configures 'eDRX-AllowedInactive' in SIB1.

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:PCYCle:EDRX:AINactive
driver.configure.signaling.nradio.cell.pcycle.edRx.ainactive.set(cell_name =
↪'abc', enable = False)
```

Configures 'eDRX-AllowedInactive' in SIB1.

param cell_name
No help available

param enable
No help available

6.3.4.10.2.418 Pcycle

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:PCYCle:PCYCle
```

class PcycleCls

Pcycle commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → PagingCycle

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:PCYCle:PCYCle
value: enums.PagingCycle = driver.configure.signaling.nradio.cell.pcycle.pcycle.
↪get(cell_name = 'abc')
```

Selects the paging cycle in radio frames.

param cell_name
No help available

return
paging_cycle: No help available

set(*cell_name: str*, *paging_cycle: PagingCycle*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:PCYCle:PCYCle
driver.configure.signaling.nradio.cell.pcycle.pcycle.set(cell_name = 'abc',␣
↪paging_cycle = enums.PagingCycle.P128)
```

Selects the paging cycle in radio frames.

param cell_name
No help available

param paging_cycle
No help available

6.3.4.10.2.419 PfOffset

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:PCYCle:PFOFfset
```

class PfOffsetCls

PfOffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Frame: enums.Frame: T1T: 1 T2 | T4 | T8 | T16: 1/2, 1/4, 1/8, 1/16
- Offset: int: No parameter help available

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:PCYCle:PFOFfset
value: GetStruct = driver.configure.signaling.nradio.cell.pcycle.pfOffset.
↪get(cell_name = 'abc')
```

Configures the field 'nAndPagingFrameOffset', used by the UE as input for the calculation of paging radio frame and paging occasion.

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *frame: Frame*, *offset: int = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:PCYCle:PFOFfset
driver.configure.signaling.nradio.cell.pcycle.pfOffset.set(cell_name = 'abc',␣
↪frame = enums.Frame.T16, offset = 1)
```

Configures the field 'nAndPagingFrameOffset', used by the UE as input for the calculation of paging radio frame and paging occasion.

param cell_name
No help available

param frame
T1T: 1 T2 | T4 | T8 | T16: 1/2, 1/4, 1/8, 1/16

param offset
No help available

6.3.4.10.2.420 PopFrame

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:PCYCle:POPFrame
```

class PopFrameCls

PopFrame commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → AntNoPorts

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:PCYCle:POPFrame
value: enums.AntNoPorts = driver.configure.signaling.nradio.cell.pcycle.
↪popFrame.get(cell_name = 'abc')
```

Configures the field 'ns', indicating the number of paging occasions per paging frame.

param cell_name
No help available

return
occasions: No help available

set(*cell_name: str*, *occasions: AntNoPorts*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:PCYCle:POPFrame
driver.configure.signaling.nradio.cell.pcycle.popFrame.set(cell_name = 'abc',␣
↪occasions = enums.AntNoPorts.P1)
```

Configures the field 'ns', indicating the number of paging occasions per paging frame.

param cell_name
No help available

param occasions
No help available

6.3.4.10.2.421 Power

class PowerCls

Power commands group definition. 45 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.power.clone()
```

Subgroups

6.3.4.10.2.422 Control

class ControlCls

Control commands group definition. 16 total commands, 8 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.power.control.clone()
```

Subgroups

6.3.4.10.2.423 Channel

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:CHANnel
```

class ChannelCls

Channel commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → SrcType

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:CHANnel
value: enums.SrcType = driver.configure.signaling.nradio.cell.power.control.
↪channel.get(cell_name = 'abc')
```

Selects the uplink channel types to which the power control commands are applied, for the initial BWP.

param cell_name
No help available

return
type_py: PUSC: PUSCH PUCC: PUCCH PUPU: PUSCH and PUCCH

set(*cell_name: str*, *type_py: SrcType*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:CHANnel
driver.configure.signaling.nradio.cell.power.control.channel.set(cell_name =
↪'abc', type_py = enums.SrcType.PUCC)
```

Selects the uplink channel types to which the power control commands are applied, for the initial BWP.

param cell_name
No help available

param type_py
PUSC: PUSCH PUCC: PUCCH PUPU: PUSCH and PUCCH

6.3.4.10.2.424 PalphaSet

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:PALPhaset
```

class PalphaSetCls

PalphaSet commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Enable: bool: ON: Signal the 'P0-PUSCH-AlphaSet'. OFF: Do not signal the 'P0-PUSCH-AlphaSet'.
- Alpha: enums.Alpha: No parameter help available
- P_0: int: No parameter help available

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:PALPhaset
value: GetStruct = driver.configure.signaling.nradio.cell.power.control.
↪palphaSet.get(cell_name = 'abc')
```

Sets the parameters 'alpha' and 'p0' of the 'P0-PUSCH-AlphaSet' that is signaled to the UE, for the initial BWP.

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *enable: bool*, *alpha: Alpha = None*, *p_0: int = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:PALPhaset
driver.configure.signaling.nradio.cell.power.control.palphaSet.set(cell_name =
↪'abc', enable = False, alpha = enums.Alpha.A00, p_0 = 1)
```

Sets the parameters 'alpha' and 'p0' of the 'P0-PUSCH-AlphaSet' that is signaled to the UE, for the initial BWP.

param cell_name
No help available

param enable
ON: Signal the 'P0-PUSCH-AlphaSet'. OFF: Do not signal the 'P0-PUSCH-AlphaSet'.

param alpha
No help available

param p_0
No help available

6.3.4.10.2.425 PbpiBpsk

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:PBPibpsk
```

class PbpiBpskCls

PbpiBpsk commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:PBPibpsk
value: bool = driver.configure.signaling.nradio.cell.power.control.pbpiBpsk.
↪get(cell_name = 'abc')
```

Configures the signaled optional Boolean value 'powerBoostPi2BPSK', for the initial BWP.

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:PBPibpsk
driver.configure.signaling.nradio.cell.power.control.pbpiBpsk.set(cell_name =
↪'abc', enable = False)
```

Configures the signaled optional Boolean value 'powerBoostPi2BPSK', for the initial BWP.

param cell_name
No help available

param enable
No help available

6.3.4.10.2.426 Pmax

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:PMAX
```

class PmaxCls

Pmax commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:PMAX
value: float or bool = driver.configure.signaling.nradio.cell.power.control.
↪pmax.get(cell_name = 'abc')
```

Defines the maximum allowed UL power 'p-Max' and whether the value is signaled to the UE (ON/value) or not (OFF) .

param cell_name
No help available

return
power: (float or boolean) No help available

set(*cell_name: str*, *power: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:PMAX
driver.configure.signaling.nradio.cell.power.control.pmax.set(cell_name = 'abc',
↪ power = 1.0)
```

Defines the maximum allowed UL power 'p-Max' and whether the value is signaled to the UE (ON/value) or not (OFF) .

param cell_name
No help available

param power
(float or boolean) No help available

6.3.4.10.2.427 PnrFr1

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:PNRFr1
```

class PnrFr1Cls

PnrFr1 commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:PNRFr1
value: float or bool = driver.configure.signaling.nradio.cell.power.control.
↪pnrFr1.get(cell_name = 'abc')
```

Parameter 'p-NR-FR1', signaled to the UE (maximum UL power allowed in the cell group across all serving cells in FR1) . Modifying this setting for a cell modifies it also for all other cells of the cell group.

param cell_name
No help available

return
power: (float or boolean) OFF: Parameter not signaled. ON: Configured value signaled.

set(*cell_name: str*, *power: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:PNRFr1
driver.configure.signaling.nradio.cell.power.control.pnrFr1.set(cell_name = 'abc
↪', power = 1.0)
```

Parameter 'p-NR-FR1', signaled to the UE (maximum UL power allowed in the cell group across all serving cells in FR1) . Modifying this setting for a cell modifies it also for all other cells of the cell group.

param cell_name
No help available

param power
(float or boolean) OFF: Parameter not signaled. ON: Configured value signaled.

6.3.4.10.2.428 PnwGrant

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:PNWGrant
```

class PnwGrantCls

PnwGrant commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:PNWGrant
value: int or bool = driver.configure.signaling.nradio.cell.power.control.
↪pnwGrant.get(cell_name = 'abc')
```

Sets the parameter 'p0-NominalWithGrant', signaled to the UE if the value is not OFF.

param cell_name
No help available

return
p_0_nomi_with_grant: (integer or boolean) No help available

set(*cell_name: str*, *p_0_nomi_with_grant: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:PNWGrant
driver.configure.signaling.nradio.cell.power.control.pnwGrant.set(cell_name =
↪'abc', p_0_nomi_with_grant = 1)
```

Sets the parameter 'p0-NominalWithGrant', signaled to the UE if the value is not OFF.

param cell_name
No help available

param p_0_nomi_with_grant
(integer or boolean) No help available

6.3.4.10.2.429 SpbPower

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:SPBPower
```

class SpbPowerCls

SpbPower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:SPBPower
value: float = driver.configure.signaling.nradio.cell.power.control.spbPower.
↪get(cell_name = 'abc')
```

Defines parameter 'SS-PBCH-BlockPower'.

param cell_name
No help available

return
power: No help available

set(*cell_name: str*, *power: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:SPBPower
driver.configure.signaling.nradio.cell.power.control.spbPower.set(cell_name =
↪'abc', power = 1.0)
```

Defines parameter 'SS-PBCH-BlockPower'.

param cell_name
No help available

param power
No help available

6.3.4.10.2.430 TpControl

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:TPControl
```

class TpControlCls

TpControl commands group definition. 9 total commands, 3 Subgroups, 1 group commands

get(*cell_name: str*) → TpControl

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:TPControl
value: enums.TpControl = driver.configure.signaling.nradio.cell.power.control.
↪tpControl.get(cell_name = 'abc')
```

Selects the pattern of TPC commands to be sent to the UE, for the initial BWP.

param cell_name
No help available

return
control: Keep, min, max, closed loop, TPC pattern, relative power tolerance.

set(*cell_name: str*, *control: TpControl*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:TPControl
driver.configure.signaling.nradio.cell.power.control.tpControl.set(cell_name =
↪'abc', control = enums.TpControl.CLOop)
```

Selects the pattern of TPC commands to be sent to the UE, for the initial BWP.

param cell_name
No help available

param control
Keep, min, max, closed loop, TPC pattern, relative power tolerance.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.power.control.tpControl.clone()
```

Subgroups

6.3.4.10.2.431 Cloop

class CloopCls

Cloop commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.power.control.tpControl.cloop.clone()
```

Subgroups

6.3.4.10.2.432 Tolerance

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:TPControl:CLOop:TOLerance
```

class ToleranceCls

Tolerance commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → float

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:TPControl:CLOop:TOLerance
value: float = driver.configure.signaling.nradio.cell.power.control.tpControl.
↪cloop.tolerance.get(cell_name = 'abc')
```

Defines the tolerance for closed-loop power control for the initial BWP.

param cell_name
No help available

return
tolerance: No help available

set(*cell_name: str*, *tolerance: float*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:TPControl:CLOop:TOLerance
driver.configure.signaling.nradio.cell.power.control.tpControl.cloop.tolerance.
↪set(cell_name = 'abc', tolerance = 1.0)
```

Defines the tolerance for closed-loop power control for the initial BWP.

param cell_name
No help available

param tolerance
No help available

6.3.4.10.2.433 Tpower

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:TPControl:CLOop:TPOWer
```

class TpowerCls

Tpower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:TPControl:CLOop:TPOWer
value: float = driver.configure.signaling.nradio.cell.power.control.tpControl.
↪cloop.tpower.get(cell_name = 'abc')
```

Defines the target power for closed-loop power control, for the initial BWP.

param cell_name
No help available

return
power: No help available

set(*cell_name: str*, *power: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:TPControl:CLOop:TPOWer
driver.configure.signaling.nradio.cell.power.control.tpControl.cloop.tpower.
↪set(cell_name = 'abc', power = 1.0)
```

Defines the target power for closed-loop power control, for the initial BWP.

param cell_name
No help available

param power
No help available

6.3.4.10.2.434 Pattern

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:TPControl:PATTern
```

class PatternCls

Pattern commands group definition. 3 total commands, 1 Subgroups, 1 group commands

get(*cell_name: str*) → TypeB

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:TPControl:PATTern
value: enums.TypeB = driver.configure.signaling.nradio.cell.power.control.
↪tpControl.pattern.get(cell_name = 'abc')
```

No command help available

param cell_name
No help available

return
type_py: No help available

set(*cell_name: str*, *type_py: TypeB*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:TPControl:PATTern
driver.configure.signaling.nradio.cell.power.control.tpControl.pattern.set(cell_
↪name = 'abc', type_py = enums.TypeB.UDEFined)
```

No command help available

param cell_name
No help available

param type_py
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.power.control.tpControl.pattern.clone()
```

Subgroups

6.3.4.10.2.435 UserDefined

class UserDefinedCls

UserDefined commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.power.control.tpControl.pattern.
↪userDefined.clone()
```

Subgroups

6.3.4.10.2.436 Mode

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:TPControl:PATTern:UDEFined:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → Repeat

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:TPControl:PATTern:UDEFined:MODE
value: enums.Repeat = driver.configure.signaling.nradio.cell.power.control.
↪tpControl.pattern.userDefined.mode.get(cell_name = 'abc')
```

Selects the mode for execution of a user-defined TPC pattern, for the initial BWP.

param cell_name
No help available

return
mode: No help available

set(*cell_name: str*, *mode: Repeat*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:TPControl:PATTern:UDEFined:MODE
driver.configure.signaling.nradio.cell.power.control.tpControl.pattern.
↪userDefined.mode.set(cell_name = 'abc', mode = enums.Repeat.CONTinuous)
```

Selects the mode for execution of a user-defined TPC pattern, for the initial BWP.

param cell_name
No help available

param mode
No help available

6.3.4.10.2.437 Pattern

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:TPControl:PATTern:UDEFined:PATTern
```

class PatternCls

Pattern commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → List[Pattern]

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:TPControl:PATTern:UDEFined:PATTern
value: List[enums.Pattern] = driver.configure.signaling.nradio.cell.power.
↪control.tpControl.pattern.userDefined.pattern.get(cell_name = 'abc')
```

Configures a user-defined TPC pattern as a sequence of commands, for the initial BWP.

param cell_name
No help available

return
pattern: Comma-separated list of commands D1: -1 dB KEEP: 0 dB U1: +1 dB U3: +3 dB A single NAV is returned if no pattern is defined.

set(*cell_name: str*, *pattern: List[Pattern]*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:TPControl:PATTern:UDEFined:PATTern
driver.configure.signaling.nradio.cell.power.control.tpControl.pattern.
↪userDefined.pattern.set(cell_name = 'abc', pattern = [Pattern.D1, Pattern.U3])
```

Configures a user-defined TPC pattern as a sequence of commands, for the initial BWP.

param cell_name
No help available

param pattern
Comma-separated list of commands D1: -1 dB KEEP: 0 dB U1: +1 dB U3: +3 dB A single NAV is returned if no pattern is defined.

6.3.4.10.2.438 RpTolerance

class RpToleranceCls

RpTolerance commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.power.control.tpControl.rpTolerance.
↪clone()
```

Subgroups

6.3.4.10.2.439 Direction

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:TPControl:RPTolerance:DIRection
```

class DirectionCls

Direction commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → TpcDirection

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:TPControl:RPTolerance:DIRection
value: enums.TpcDirection = driver.configure.signaling.nradio.cell.power.
↪control.tpControl.rpTolerance.direction.get(cell_name = 'abc')
```

Selects the direction of the TPC pattern for relative power tolerance tests, for the initial BWP.

param cell_name
No help available

return
direction: No help available

set(*cell_name: str*, *direction: TpcDirection*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:TPControl:RPTolerance:DIRection
driver.configure.signaling.nradio.cell.power.control.tpControl.rpTolerance.
↪direction.set(cell_name = 'abc', direction = enums.TpcDirection.ALTernating)
```

Selects the direction of the TPC pattern for relative power tolerance tests, for the initial BWP.

param cell_name
No help available

param direction
No help available

6.3.4.10.2.440 Pattern

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:TPControl:RPTolerance:PATTern
```

class PatternCls

Pattern commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → RpPattern

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:TPControl:RPTolerance:PATTern
value: enums.RpPattern = driver.configure.signaling.nradio.cell.power.control.
↪tpControl.rpTolerance.pattern.get(cell_name = 'abc')
```

Selects a TPC pattern for ramping up and ramping down relative power tolerance tests, for the initial BWP.

param cell_name
No help available

return
pattern: No help available

set(*cell_name: str*, *pattern: RpPattern*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:TPControl:RPTolerance:PATTern
driver.configure.signaling.nradio.cell.power.control.tpControl.rpTolerance.
↪pattern.set(cell_name = 'abc', pattern = enums.RpPattern.A)
```

Selects a TPC pattern for ramping up and ramping down relative power tolerance tests, for the initial BWP.

param cell_name
No help available

param pattern
No help available

6.3.4.10.2.441 StId

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:TPControl:RPTolerance:STID
```

class StIdCls

StId commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:TPControl:RPTolerance:STID
value: int = driver.configure.signaling.nradio.cell.power.control.tpControl.
↪rpTolerance.stId.get(cell_name = 'abc')
```

Selects the subtest ID for relative power tolerance tests, for the initial BWP.

param cell_name
No help available

return
sub_test_id: No help available

set(*cell_name: str*, *sub_test_id: int*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:POWer:CONTrol:TPControl:RPTolerance:STID
driver.configure.signaling.nradio.cell.power.control.tpControl.rpTolerance.stId.
↪set(cell_name = 'abc', sub_test_id = 1)
```

Selects the subtest ID for relative power tolerance tests, for the initial BWP.

param cell_name
No help available

param sub_test_id
No help available

6.3.4.10.2.442 Downlink

class DownlinkCls

Downlink commands group definition. 15 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.power.downlink.clone()
```

Subgroups

6.3.4.10.2.443 Ocng

class OcngCls

Ocng commands group definition. 8 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.power.downlink.ocng.clone()
```

Subgroups

6.3.4.10.2.444 Enable

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:OCNG:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:OCNG:ENABle
value: bool = driver.configure.signaling.nradio.cell.power.downlink.ocng.enable.
↪get(cell_name = 'abc')
```

Enables or disables the OFDMA channel noise generator (OCNG) .

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:OCNG:ENABle
driver.configure.signaling.nradio.cell.power.downlink.ocng.enable.set(cell_name␣
↪= 'abc', enable = False)
```

Enables or disables the OFDMA channel noise generator (OCNG) .

param cell_name
No help available

param enable
No help available

6.3.4.10.2.445 Pdcch

class PdcchCls

Pdcch commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.power.downlink.ocng.pdcch.clone()
```

Subgroups

6.3.4.10.2.446 DscSsb

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:OCNG:PDCCh:DSCSsb
```

class DscSsbCls

DscSsb commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:OCNG:PDCCh:DSCSsb
value: bool = driver.configure.signaling.nradio.cell.power.downlink.ocng.pdcch.
↪dscSsb.get(cell_name = 'abc')
```

Disables OCNG for PDCCH in slots carrying SSB.

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:OCNG:PDCCh:DSCSsb
driver.configure.signaling.nradio.cell.power.downlink.ocng.pdcch.dscSsb.
↪set(cell_name = 'abc', enable = False)
```

Disables OCNG for PDCCH in slots carrying SSB.

param cell_name
No help available

param enable
No help available

6.3.4.10.2.447 Poffset

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:OCNG:PDCCh:POFFset
```

class PoffsetCls

Poffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Mode: enums.ModeD: No parameter help available
- Value: float: Power level relative to the SSB EPRE.

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:OCNG:PDCCh:POFFset
value: GetStruct = driver.configure.signaling.nradio.cell.power.downlink.ocng.
↪pdcch.poffset.get(cell_name = 'abc')
```

Defines the power level of the PDCCH for OCNG.

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *mode: ModeD = None*, *value: float = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:OCNG:PDCCh:POFFset
driver.configure.signaling.nradio.cell.power.downlink.ocng.pdcch.poffset.
↪set(cell_name = 'abc', mode = enums.ModeD.MAX, value = 1.0)
```

Defines the power level of the PDCCH for OCNG.

param cell_name
No help available

param mode
No help available

param value
Power level relative to the SSB EPRE.

6.3.4.10.2.448 Rb

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:OCNG:PDCCh:RB
```

class RbCls

Rb commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Nrb: int: No parameter help available
- Start_Rb: int: No parameter help available

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:OCNG:PDCCh:RB
value: GetStruct = driver.configure.signaling.nradio.cell.power.downlink.ocng.
↪pdcch.rb.get(cell_name = 'abc')
```

No command help available

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *nrb: int*, *start_rb: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:OCNG:PDCCh:RB
driver.configure.signaling.nradio.cell.power.downlink.ocng.pdcch.rb.set(cell_
↪name = 'abc', nrb = 1, start_rb = 1)
```

No command help available

param cell_name
No help available

param nrb
No help available

param start_rb
No help available

6.3.4.10.2.449 Pdsch

class PdschCls

Pdsch commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.power.downlink.ocng.pdsch.clone()
```

Subgroups

6.3.4.10.2.450 DscSsb

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:OCNG:PDSCh:DSCSsb
```

class DscSsbCls

DscSsb commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:OCNG:PDSCh:DSCSsb
value: bool = driver.configure.signaling.nradio.cell.power.downlink.ocng.pdsch.
↪dscSsb.get(cell_name = 'abc')
```

Disables OCNG for PDSCH in slots carrying SSB.

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:OCNG:PDSCh:DSCSsb
driver.configure.signaling.nradio.cell.power.downlink.ocng.pdsch.dscSsb.
↪set(cell_name = 'abc', enable = False)
```

Disables OCNG for PDSCH in slots carrying SSB.

param cell_name
No help available

param enable
No help available

6.3.4.10.2.451 Modulation

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:OCNG:PDSCh:MODulation
```

class ModulationCls

Modulation commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → Modulation

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:OCNG:PDSCh:MODulation
value: enums.Modulation = driver.configure.signaling.nradio.cell.power.downlink.
↪ocng.pdsch.modulation.get(cell_name = 'abc')
```

Selects the modulation scheme of the PDSCH for OCNG.

param cell_name
No help available

return
modulation: No help available

set(*cell_name: str*, *modulation: Modulation*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:OCNG:PDSCh:MODulation
driver.configure.signaling.nradio.cell.power.downlink.ocng.pdsch.modulation.
↪set(cell_name = 'abc', modulation = enums.Modulation.BPSK)
```

Selects the modulation scheme of the PDSCH for OCNG.

param cell_name
No help available

param modulation
No help available

6.3.4.10.2.452 Poffset

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:OCNG:PDSCh:POFFset
```

class PoffsetCls

Poffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Mode: enums.ModeD: No parameter help available
- Value: float: Power level relative to the SSB EPRE.

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:OCNG:PDSCh:POFFset
value: GetStruct = driver.configure.signaling.nradio.cell.power.downlink.ocng.
↪pdsch.poffset.get(cell_name = 'abc')
```

Defines the power level of the PDSCH for OCNG.

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *mode: ModeD = None*, *value: float = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:OCNG:PDSCh:POFFset
driver.configure.signaling.nradio.cell.power.downlink.ocng.pdsch.poffset.
↪set(cell_name = 'abc', mode = enums.ModeD.MAX, value = 1.0)
```

Defines the power level of the PDSCH for OCNG.

param cell_name
No help available

param mode
No help available

param value
Power level relative to the SSB EPRE.

6.3.4.10.2.453 Rb

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:OCNG:PDSCh:RB
```

class RbCls

Rb commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Nrb: int: No parameter help available
- Start_Rb: int: No parameter help available

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:OCNG:PDSCh:RB
value: GetStruct = driver.configure.signaling.nradio.cell.power.downlink.ocng.
↪pdsch.rb.get(cell_name = 'abc')
```

No command help available

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *nrb: int*, *start_rb: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:OCNG:PDSCh:RB
driver.configure.signaling.nradio.cell.power.downlink.ocng.pdsch.rb.set(cell_
↪name = 'abc', nrb = 1, start_rb = 1)
```

No command help available

param cell_name
No help available

param nrb
No help available

param start_rb
No help available

6.3.4.10.2.454 Poffset

class PoffsetCls

Poffset commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.power.downlink.poffset.clone()
```

Subgroups

6.3.4.10.2.455 Coreset

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:POFFset:COReset
```

class CoresetCls

Coreset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:POFFset:COReset
value: float = driver.configure.signaling.nradio.cell.power.downlink.poffset.
↪coreset.get(cell_name = 'abc')
```

Defines the offset of the CORESET power relative to the SSS power.

param cell_name
No help available

return
coreset: No help available

set(*cell_name: str*, *coreset: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:POFFset:COReset
driver.configure.signaling.nradio.cell.power.downlink.poffset.coreset.set(cell_
↪name = 'abc', coreset = 1.0)
```

Defines the offset of the CORESET power relative to the SSS power.

param cell_name
No help available

param coreset
No help available

6.3.4.10.2.456 NrDl

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:POFFset:NRDL
```

class NrDlCls

NrDl commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:POFFset:NRDL
value: float = driver.configure.signaling.nradio.cell.power.downlink.poffset.
↪nrDl.get(cell_name = 'abc')
```

Defines the offset of the DL power (PDSCH) relative to the SSS power.

param cell_name
No help available

return
nr_dl: No help available

set(*cell_name: str*, *nr_dl: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:POFFset:NRDL
driver.configure.signaling.nradio.cell.power.downlink.poffset.nrDl.set(cell_
↪name = 'abc', nr_dl = 1.0)
```

Defines the offset of the DL power (PDSCH) relative to the SSS power.

param cell_name
No help available

param nr_dl
No help available

6.3.4.10.2.457 Pss

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:POFFset:PSS
```

class PssCls

Pss commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:POFFset:PSS
value: int = driver.configure.signaling.nradio.cell.power.downlink.poffset.pss.
↪get(cell_name = 'abc')
```

Defines the offset of the PSS power relative to the SSS power.

param cell_name
No help available

return
pss: 0: PSS EPRE = SSS EPRE 3: PSS EPRE = SSS EPRE + 3 dB

set(*cell_name: str*, *pss: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:POFFset:PSS
driver.configure.signaling.nradio.cell.power.downlink.poffset.pss.set(cell_name␣
↪= 'abc', pss = 1)
```

Defines the offset of the PSS power relative to the SSS power.

param cell_name
No help available

param pss
0: PSS EPRE = SSS EPRE 3: PSS EPRE = SSS EPRE + 3 dB

6.3.4.10.2.458 Ssb

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:POFFset:SSB
```

class SsbCls

Ssb commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:POFFset:SSB
value: float = driver.configure.signaling.nradio.cell.power.downlink.poffset.
↪ssb.get(cell_name = 'abc')
```

Defines the offset of the SSB power relative to the beam power.

param cell_name
No help available

return
ssb: No help available

set(*cell_name: str*, *ssb: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:POFFset:SSB
driver.configure.signaling.nradio.cell.power.downlink.poffset.ssb.set(cell_name␣
↪= 'abc', ssb = 1.0)
```

Defines the offset of the SSB power relative to the beam power.

param cell_name
No help available

param ssb
No help available

6.3.4.10.2.459 PppScaling

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:PPPScaling
```

class PppScalingCls

PppScaling commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → PowerScaling

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:PPPScaling
value: enums.PowerScaling = driver.configure.signaling.nradio.cell.power.
↪downlink.pppScaling.get(cell_name = 'abc')
```

Defines the PDSCH power scaling depending on the number of layers.

param cell_name
No help available

return
power_scaling: - TGPP: 3GPP compliant - the power per layer decreases with increasing number of layers, so that the total power of the PDSCH over all layers is the same as for single-layer transmission. - TOPTimized: Throughput optimized - the total power of the PDSCH increases with increasing number of layers (double number of layers means plus 3 dB) .

set(*cell_name: str*, *power_scaling: PowerScaling*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:PPPScaling
driver.configure.signaling.nradio.cell.power.downlink.pppScaling.set(cell_name␣
↪= 'abc', power_scaling = enums.PowerScaling.TGPP)
```

Defines the PDSCH power scaling depending on the number of layers.

param cell_name
No help available

param power_scaling

- TGPP: 3GPP compliant - the power per layer decreases with increasing number of layers, so that the total power of the PDSCH over all layers is the same as for single-layer transmission.
- TOPTimized: Throughput optimized - the total power of the PDSCH increases with increasing number of layers (double number of layers means plus 3 dB) .

6.3.4.10.2.460 Sepre

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:SEPRe
```

class SepreCls

Sepre commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:SEPRe
value: float = driver.configure.signaling.nradio.cell.power.downlink.sepre.
↪get(cell_name = 'abc')
```

Defines the SSB EPRE.

param cell_name
No help available

return
epre: No help available

set(*cell_name: str*, *epre: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:SEPRe
driver.configure.signaling.nradio.cell.power.downlink.sepre.set(cell_name = 'abc
↪', epre = 1.0)
```

Defines the SSB EPRE.

param cell_name
No help available

param epre
No help available

6.3.4.10.2.461 Tcell

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:TCELl
```

class TcellCls

Tcell commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:TCELl
value: float = driver.configure.signaling.nradio.cell.power.downlink.tcell.
↪get(cell_name = 'abc')
```

Defines the total cell power.

param cell_name
No help available

return
power: No help available

set(*cell_name: str*, *power: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:TCELl
driver.configure.signaling.nradio.cell.power.downlink.tcell.set(cell_name = 'abc
↪', power = 1.0)
```

Defines the total cell power.

param cell_name
No help available

param power
No help available

6.3.4.10.2.462 Uplink

class UplinkCls

Uplink commands group definition. 14 total commands, 13 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.power.uplink.clone()
```

Subgroups

6.3.4.10.2.463 Auto

class AutoCls

Auto commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.power.uplink.auto.clone()
```

Subgroups

6.3.4.10.2.464 RlOffset

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:AUTO:RLOFfset
```

class RlOffsetCls

RlOffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:AUTO:RLOFfset
value: float = driver.configure.signaling.nradio.cell.power.uplink.auto.
↪rlOffset.get(cell_name = 'abc')
```

Sets the reference level offset for automatic configuration of expected UL power.

param cell_name
No help available

return
ref_level_offset: No help available

set(*cell_name: str*, *ref_level_offset: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:AUTO:RLOFfset
driver.configure.signaling.nradio.cell.power.uplink.auto.rlOffset.set(cell_name␣
↪= 'abc', ref_level_offset = 1.0)
```

Sets the reference level offset for automatic configuration of expected UL power.

param cell_name
No help available

param ref_level_offset
No help available

6.3.4.10.2.465 Rsource

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:AUTO:RSOurce
```

class RsourceCls

Rsource commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → SrcType

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:AUTO:RSOurce
value: enums.SrcType = driver.configure.signaling.nradio.cell.power.uplink.auto.
↪rsource.get(cell_name = 'abc')
```

Sets the reference source for automatic configuration of expected UL power.

param cell_name
No help available

return
ref_source: PUSC: PUSCH PUCC: PUCCH PUPU: PUCCH and PUSCH

set(*cell_name: str*, *ref_source: SrcType*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:AUTO:RSOurce
driver.configure.signaling.nradio.cell.power.uplink.auto.rsource.set(cell_name
↪= 'abc', ref_source = enums.SrcType.PUCC)
```

Sets the reference source for automatic configuration of expected UL power.

param cell_name
No help available

param ref_source
PUSC: PUSCH PUCC: PUCCH PUPU: PUCCH and PUSCH

6.3.4.10.2.466 Cindex

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:CINDex
```

class CindexCls

Cindex commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:CINDex
value: int = driver.configure.signaling.nradio.cell.power.uplink.cindex.
↪get(cell_name = 'abc')
```

Sets the PRACH configuration index to be used by the UE.

param cell_name
No help available

return
index: No help available

set(*cell_name: str*, *index: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:CINDex
driver.configure.signaling.nradio.cell.power.uplink.cindex.set(cell_name = 'abc
↪', index = 1)
```

Sets the PRACH configuration index to be used by the UE.

param cell_name
No help available

param index
No help available

6.3.4.10.2.467 IpPreambles

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:IPPReambles
```

class IpPreamblesCls

IpPreambles commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Ignore_Prach_Mode: enums.IgnorePrachMode: IALLways: ignore all preambles IXTimes: ignore NoIgnored preambles RALLways: respond to all preambles
- No_Ignored: int: Number of preambles to be ignored for IgnorePrachMode = IXTimes.

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:IPPReambles
value: GetStruct = driver.configure.signaling.nradio.cell.power.uplink.
↪ipPreambles.get(cell_name = 'abc')
```

Selects the behavior of the signaling application when receiving preambles from the UE. The setting is synchronized over all NR cells (identical values for all NR cells) .

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *ignore_prach_mode: IgnorePrachMode*, *no_ignored: int = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:IPPReambles
driver.configure.signaling.nradio.cell.power.uplink.ipPreambles.set(cell_name =
↪'abc', ignore_prach_mode = enums.IgnorePrachMode.IALLways, no_ignored = 1)
```

Selects the behavior of the signaling application when receiving preambles from the UE. The setting is synchronized over all NR cells (identical values for all NR cells) .

param cell_name
No help available

param ignore_prach_mode
IALLways: ignore all preambles IXTimes: ignore NoIgnored preambles RALLways: respond to all preambles

param no_ignored
Number of preambles to be ignored for IgnorePrachMode = IXTimes.

6.3.4.10.2.468 LrsIndex

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:LRSindex
```

class LrsIndexCls

LrsIndex commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:LRSindex
value: int = driver.configure.signaling.nradio.cell.power.uplink.lrsIndex.
↪get(cell_name = 'abc')
```

Sets the parameter 'prach-RootSequenceIndex', signaled to the UE.

param cell_name
No help available

return
index: No help available

set(*cell_name: str*, *index: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:LRSindex
driver.configure.signaling.nradio.cell.power.uplink.lrsIndex.set(cell_name =
↪'abc', index = 1)
```

Sets the parameter 'prach-RootSequenceIndex', signaled to the UE.

param cell_name
No help available

param index
No help available

6.3.4.10.2.469 Meepre

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:MEEPre
```

class MeepreCls

Meepre commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:MEEPre
value: float = driver.configure.signaling.nradio.cell.power.uplink.meepre.
↪get(cell_name = 'abc')
```

Defines the maximum EPRE expected in the UL, for user-defined configuration. For automatic configuration, you can query the value.

param cell_name
No help available

return
power: No help available

set(*cell_name: str*, *power: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:MEEPre
driver.configure.signaling.nradio.cell.power.uplink.meepre.set(cell_name = 'abc
↪', power = 1.0)
```

Defines the maximum EPRE expected in the UL, for user-defined configuration. For automatic configuration, you can query the value.

param cell_name
No help available

param power
No help available

6.3.4.10.2.470 MeRms

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:MERMs
```

class MeRmsCls

MeRms commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:MERMs
value: float = driver.configure.signaling.nradio.cell.power.uplink.meRms.
↪get(cell_name = 'abc')
```

Defines the maximum expected RMS UL power, for user-defined configuration. For automatic configuration, you can query the value.

param cell_name
No help available

return
power: No help available

set(*cell_name: str*, *power: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:MERMs
driver.configure.signaling.nradio.cell.power.uplink.meRms.set(cell_name = 'abc',
↪ power = 1.0)
```

Defines the maximum expected RMS UL power, for user-defined configuration. For automatic configuration, you can query the value.

param cell_name
No help available

param power
No help available

6.3.4.10.2.471 Npreambles

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:NPReambles
```

class NpreamblesCls

Npreambles commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:NPReambles
value: int = driver.configure.signaling.nradio.cell.power.uplink.npreambles.
↪get(cell_name = 'abc')
```

Sets the parameter 'numberOfPreamblesPerSSB-ForThisPartition-r17', signaled in 'FeatureCombinationPreambles-r17' to the UE.

param cell_name
No help available

return
no_preambles: No help available

set(*cell_name: str*, *no_preambles: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:NPReambles
driver.configure.signaling.nradio.cell.power.uplink.npreambles.set(cell_name =
↪'abc', no_preambles = 1)
```

Sets the parameter 'numberOfPreamblesPerSSB-ForThisPartition-r17', signaled in 'FeatureCombinationPreambles-r17' to the UE.

param cell_name
No help available

param no_preambles
No help available

6.3.4.10.2.472 PrStep

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:PRSTep
```

class PrStepCls

PrStep commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → PwrRampingStepB

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:PRSTep
value: enums.PwrRampingStepB = driver.configure.signaling.nradio.cell.power.
↪uplink.prStep.get(cell_name = 'abc')
```

Defines the transmit power difference between two consecutive preambles (power ramping) .

param cell_name
No help available

return
pwr_ramping_step: Step size in dB (0 dB to 4 dB)

set(*cell_name: str*, *pwr_ramping_step: PwrRampingStepB*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:PRSTep
driver.configure.signaling.nradio.cell.power.uplink.prStep.set(cell_name = 'abc
↪', pwr_ramping_step = enums.PwrRampingStepB.S0)
```

Defines the transmit power difference between two consecutive preambles (power ramping) .

param cell_name
No help available

param pwr_ramping_step
Step size in dB (0 dB to 4 dB)

6.3.4.10.2.473 PrtPower

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:PRTPower
```

class PrtPowerCls

PrtPower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:PRTPower
value: int = driver.configure.signaling.nradio.cell.power.uplink.prtPower.
↪get(cell_name = 'abc')
```

Sets the parameter 'preambleReceivedTargetPower', signaled to the UE.

param cell_name
No help available

return
power: No help available

set(*cell_name: str*, *power: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:PRTPower
driver.configure.signaling.nradio.cell.power.uplink.prtPower.set(cell_name =
↪'abc', power = 1)
```

Sets the parameter 'preambleReceivedTargetPower', signaled to the UE.

param cell_name
No help available

param power
No help available

6.3.4.10.2.474 RcMode

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:RCMode
```

class RcModeCls

RcMode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → ModeB

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:RCMode
value: enums.ModeB = driver.configure.signaling.nradio.cell.power.uplink.rcMode.
↪get(cell_name = 'abc')
```

Selects a configuration mode for the expected UL level.

param cell_name
No help available

return
mode: Automatic configuration or user-defined configuration

set(*cell_name: str*, *mode: ModeB*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:RCMode
driver.configure.signaling.nradio.cell.power.uplink.rcMode.set(cell_name = 'abc
↪', mode = enums.ModeB.AUTO)
```

Selects a configuration mode for the expected UL level.

param cell_name
No help available

param mode
Automatic configuration or user-defined configuration

6.3.4.10.2.475 RedCap

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:REDCap
```

class RedCapCls

RedCap commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:REDCap
value: bool = driver.configure.signaling.nradio.cell.power.uplink.redCap.
↪get(cell_name = 'abc')
```

Sets the parameter 'redCap-r17', signaled in 'FeatureCombination-r17' to the UE.

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:REDCap
driver.configure.signaling.nradio.cell.power.uplink.redCap.set(cell_name = 'abc
↪', enable = False)
```

Sets the parameter 'redCap-r17', signaled in 'FeatureCombination-r17' to the UE.

param cell_name
No help available

param enable
No help available

6.3.4.10.2.476 Spreambles

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:SPReambles
```

class SpreamblesCls

Spreambles commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:SPReambles
value: int = driver.configure.signaling.nradio.cell.power.uplink.spreambles.
↪get(cell_name = 'abc')
```

Sets the parameter 'startPreambleForThisPartition-r17', signaled in 'FeatureCombinationPreambles-r17' to the UE.

param cell_name
No help available

return
start_preambles: No help available

set(*cell_name: str*, *start_preambles: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:SPReambles
driver.configure.signaling.nradio.cell.power.uplink.spreambles.set(cell_name =
↪'abc', start_preambles = 1)
```

Sets the parameter 'startPreambleForThisPartition-r17', signaled in 'FeatureCombinationPreambles-r17' to the UE.

param cell_name
No help available

param start_preambles
No help available

6.3.4.10.2.477 ZczConfig

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:ZCZConfig
```

class ZczConfigCls

ZczConfig commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:ZCZConfig
value: int = driver.configure.signaling.nradio.cell.power.uplink.zczConfig.
↪get(cell_name = 'abc')
```

Sets the parameter 'zeroCorrelationZoneConfig', signaled to the UE.

param cell_name
No help available

return
config: No help available

set(*cell_name: str*, *config: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:ZCZConfig
driver.configure.signaling.nradio.cell.power.uplink.zczConfig.set(cell_name =
↪'abc', config = 1)
```

Sets the parameter 'zeroCorrelationZoneConfig', signaled to the UE.

param cell_name
No help available

param config
No help available

6.3.4.10.2.478 Pucch

class PucchCls

Pucch commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.pucch.clone()
```

Subgroups

6.3.4.10.2.479 FormatPy

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:PUCCh:FORMat
```

class FormatPyCls

FormatPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → CellPucchFormatPy

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:PUCCh:FORMat
value: enums.CellPucchFormatPy = driver.configure.signaling.nradio.cell.pucch.
↪formatPy.get(cell_name = 'abc')
```

Selects the PUCCH format for the initial BWP.

param cell_name
No help available

return
format_py: No help available

set(*cell_name: str*, *format_py: CellPucchFormatPy*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:PUCCh:FORMat
driver.configure.signaling.nradio.cell.pucch.formatPy.set(cell_name = 'abc',␣
↪format_py = enums.CellPucchFormatPy.F0)
```

Selects the PUCCH format for the initial BWP.

param cell_name
No help available

param format_py
No help available

6.3.4.10.2.480 Mode

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:PUCCh:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → ConfigMode

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:PUCCh:MODE
value: enums.ConfigMode = driver.configure.signaling.nradio.cell.pucch.mode.
↪get(cell_name = 'abc')
```

Selects the configuration mode for the PUCCH resources, for the initial BWP.

param cell_name
No help available

return
mode: No help available

set(*cell_name: str*, *mode: ConfigMode*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:PUCCh:MODE
driver.configure.signaling.nradio.cell.pucch.mode.set(cell_name = 'abc', mode =␣
↪enums.ConfigMode.AUTO)
```

Selects the configuration mode for the PUCCH resources, for the initial BWP.

param cell_name
No help available

param mode
No help available

6.3.4.10.2.481 Papr

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:PUCCh:PAPR
```

class PaprCls

Papr commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:PUCCh:PAPR
value: bool = driver.configure.signaling.nradio.cell.pucch.papr.get(cell_name =
↪'abc')
```

Enables the usage of a DMRS with a low PAPR, for PUCCH, initial BWP.

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:PUCCh:PAPR
driver.configure.signaling.nradio.cell.pucch.papr.set(cell_name = 'abc', enable␣
↪= False)
```

Enables the usage of a DMRS with a low PAPR, for PUCCH, initial BWP.

param cell_name
No help available

param enable
No help available

6.3.4.10.2.482 Sprb

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:PUCCh:SPRB
```

class SprbCls

Sprb commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → LowHigh

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:PUCCh:SPRB
value: enums.LowHigh = driver.configure.signaling.nradio.cell.pucch.sprb.
↪get(cell_name = 'abc')
```

Selects the position of the resource blocks: lower end or upper end of the allowed range. For the initial BWP.

param cell_name
No help available

return
starting_prb: No help available

set(*cell_name: str*, *starting_prb: LowHigh*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:PUCCh:SPRB
driver.configure.signaling.nradio.cell.pucch.sprb.set(cell_name = 'abc',␣
↪starting_prb = enums.LowHigh.HIGH)
```

Selects the position of the resource blocks: lower end or upper end of the allowed range. For the initial BWP.

param cell_name
No help available

param starting_prb
No help available

6.3.4.10.2.483 Pusch

class PuschCls

Pusch commands group definition. 6 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.pusch.clone()
```

Subgroups

6.3.4.10.2.484 Dtfs

class DtfsCls

Dtfs commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.pusch.dtfs.clone()
```

Subgroups

6.3.4.10.2.485 McsTable

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:PUSCh:DTFS:MCSTable
```

class McsTableCls

McsTable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → McsTableB

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:PUSCh:DTFS:MCSTable
value: enums.McsTableB = driver.configure.signaling.nradio.cell.pusch.dtfs.
↪mcsTable.get(cell_name = 'abc')
```

Defines which MCS table must be used for PUSCH with transform precoding (parameter 'mcs-TableTransformPrecoder') , for the initial BWP.

param cell_name
No help available

return
mcs: 256QAM, 64QAM low SE, 64QAM

set(*cell_name: str*, *mcs: McsTableB*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:PUSCh:DTFS:MCSTable
driver.configure.signaling.nradio.cell.pusch.dtfs.mcsTable.set(cell_name = 'abc
↪', mcs = enums.McsTableB.L64)
```

Defines which MCS table must be used for PUSCH with transform precoding (parameter 'mcsTableTransformPrecoder') , for the initial BWP.

param cell_name
No help available

param mcs
256QAM, 64QAM low SE, 64QAM

6.3.4.10.2.486 PibPsk

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:PUSCh:DTFS:PIBPsk
```

class PibPskCls

PibPsk commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:PUSCh:DTFS:PIBPsk
value: bool = driver.configure.signaling.nradio.cell.pusch.dtfs.pibPsk.get(cell_
↪name = 'abc')
```

Enables the modulation scheme /2-BPSK for PUSCH with transform precoding, for the initial BWP.

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:PUSCh:DTFS:PIBPsk
driver.configure.signaling.nradio.cell.pusch.dtfs.pibPsk.set(cell_name = 'abc',␣
↪enable = False)
```

Enables the modulation scheme /2-BPSK for PUSCH with transform precoding, for the initial BWP.

param cell_name
No help available

param enable
No help available

6.3.4.10.2.487 TpRecoding

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:PUSCh:TPRecoding
```

class TpRecodingCls

TpRecoding commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → Waveform

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:PUSCh:TPRecoding
value: enums.Waveform = driver.configure.signaling.nradio.cell.pusch.tpRecoding.
↪get(cell_name = 'abc')
```

Defines which type of OFDM the UE must use for the PUSCH, for the initial BWP.

param cell_name
No help available

return
waveform: CP: CP-OFDM (no transform precoding) . DTFS: DFT-s-OFDM (with transform precoding) .

set(*cell_name: str*, *waveform: Waveform*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:PUSCh:TPRecoding
driver.configure.signaling.nradio.cell.pusch.tpRecoding.set(cell_name = 'abc',␣
↪waveform = enums.Waveform.CP)
```

Defines which type of OFDM the UE must use for the PUSCH, for the initial BWP.

param cell_name
No help available

param waveform
CP: CP-OFDM (no transform precoding) . DTFS: DFT-s-OFDM (with transform precoding) .

6.3.4.10.2.488 Tschema

class TschemaCls

Tschema commands group definition. 3 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.pusch.tschema.clone()
```

Subgroups

6.3.4.10.2.489 Codebook

class CodebookCls

Codebook commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.pusch.tschema.codebook.clone()
```

Subgroups

6.3.4.10.2.490 FptMode

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:PUSCh:TSCHema:CODebook:FPTMode
```

class FptModeCls

FptMode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → FtpMode

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:PUSCh:TSCHema:CODebook:FPTMode
value: enums.FtpMode = driver.configure.signaling.nradio.cell.pusch.tschema.
↪codebook.fptMode.get(cell_name = 'abc')
```

Selects the signaled 'ul-FullPowerTransmission-r16', for the initial BWP.

param cell_name
No help available

return
mode: AUTO: signaled value selected via reported UE capabilities FULL: signaled value 'fullpower' MOD1: signaled value 'fullpowerMode1' MOD2: signaled value 'fullpowerMode2' OFF: 'ul-FullPowerTransmission-r16' not signaled

set(*cell_name: str*, *mode: FtpMode*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:PUSCh:TSCHema:CODebook:FPTMode
driver.configure.signaling.nradio.cell.pusch.tschema.codebook.fptMode.set(cell_
↪name = 'abc', mode = enums.FtpMode.AUTO)
```

Selects the signaled 'ul-FullPowerTransmission-r16', for the initial BWP.

param cell_name
No help available

param mode
AUTO: signaled value selected via reported UE capabilities FULL: signaled value 'fullpower' MOD1: signaled value 'fullpowerMode1' MOD2: signaled value 'fullpowerMode2' OFF: 'ul-FullPowerTransmission-r16' not signaled

6.3.4.10.2.491 Subset

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:PUSCh:TSCHema:CODebook:SUBSet
```

class SubsetCls

Subset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → CodebookSubset

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:PUSCh:TSCHema:CODebook:SUBSet
value: enums.CodebookSubset = driver.configure.signaling.nradio.cell.pusch.
↪tschema.codebook.subset.get(cell_name = 'abc')
```

Selects the codebook subset for codebook-based transmission (signaled 'codebookSubset') , for the initial BWP.

param cell_name
No help available

return
subset: AUTO: signaled value selected via reported UE capabilities FPNC: signaled value 'fullyAndPartialAndNonCoherent' PNC: signaled value 'partialAndNonCoherent', currently not supported NC: signaled value 'nonCoherent'

set(*cell_name: str*, *subset: CodebookSubset*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:PUSCh:TSCHema:CODebook:SUBSet
driver.configure.signaling.nradio.cell.pusch.tschema.codebook.subset.set(cell_
↪name = 'abc', subset = enums.CodebookSubset.AUTO)
```

Selects the codebook subset for codebook-based transmission (signaled 'codebookSubset') , for the initial BWP.

param cell_name
No help available

param subset
AUTO: signaled value selected via reported UE capabilities FPNC: signaled value 'fullyAndPartialAndNonCoherent' PNC: signaled value 'partialAndNonCoherent', currently not supported NC: signaled value 'nonCoherent'

6.3.4.10.2.492 Mode

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:PUSCh:TSCHema:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → Choice

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:PUSCh:TSCHema:MODE
value: enums.Choice = driver.configure.signaling.nradio.cell.pusch.tschema.mode.
↪get(cell_name = 'abc')
```

Selects the PUSCH transmission scheme, signaled as 'txConfig', for the initial BWP.

param cell_name
No help available

return
choice: SINGle: single antenna port, 'txConfig' not signaled CODebook: codebook-based transmission NCODebook: currently not supported

set(*cell_name: str*, *choice: Choice*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:PUSCh:TSCHema:MODE
driver.configure.signaling.nradio.cell.pusch.tschema.mode.set(cell_name = 'abc',
↪ choice = enums.Choice.CODebook)
```

Selects the PUSCH transmission scheme, signaled as 'txConfig', for the initial BWP.

param cell_name
No help available

param choice
SINGle: single antenna port, 'txConfig' not signaled CODebook: codebook-based transmission NCODebook: currently not supported

6.3.4.10.2.493 ReSelection

class ReSelectionCls

ReSelection commands group definition. 11 total commands, 6 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.reSelection.clone()
```

Subgroups

6.3.4.10.2.494 Common

class CommonCls

Common commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.reSelection.common.clone()
```

Subgroups

6.3.4.10.2.495 Qhyst

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:RESelection:COMMon:QHYSt
```

class QhystCls

Qhyst commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RESelection:COMMon:QHYSt
value: int = driver.configure.signaling.nradio.cell.reSelection.common.qhyst.
↪get(cell_name = 'abc')
```

Configures the parameter 'q-Hyst', signaled to the UE in SIB2.

param cell_name
No help available

return
power: No help available

set(*cell_name: str*, *power: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RESelection:COMMon:QHYSt
driver.configure.signaling.nradio.cell.reSelection.common.qhyst.set(cell_name =
↪'abc', power = 1)
```

Configures the parameter 'q-Hyst', signaled to the UE in SIB2.

param cell_name
No help available

param power
No help available

6.3.4.10.2.496 Range

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:RESelection:COMMon:RANGe
```

class RangeCls

Range commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RESelection:COMMon:RANGe
value: int or bool = driver.configure.signaling.nradio.cell.reSelection.common.
↪range.get(cell_name = 'abc')
```

Configures the parameter 'RangeToBestCell', signaled to the UE in SIB2 if the value is not OFF

param cell_name
No help available

return
power: (integer or boolean) No help available

set(*cell_name: str*, *power: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RESelection:COMMon:RANGe
driver.configure.signaling.nradio.cell.reSelection.common.range.set(cell_name =
↪'abc', power = 1)
```

Configures the parameter 'RangeToBestCell', signaled to the UE in SIB2 if the value is not OFF

param cell_name
No help available

param power
(integer or boolean) No help available

6.3.4.10.2.497 MinLevel

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:RESelection:MINLevel
```

class MinLevelCls

MinLevel commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RESelection:MINLevel
value: int or bool = driver.configure.signaling.nradio.cell.reSelection.
↪minLevel.get(cell_name = 'abc')
```

Configures the parameter 'q-RxLevMinSUL', signaled to the UE in SIB2 if the value is not OFF.

param cell_name
No help available

return
power: (integer or boolean) No help available

set(*cell_name: str*, *power: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RESelection:MINLevel
driver.configure.signaling.nradio.cell.reSelection.minLevel.set(cell_name = 'abc
↪', power = 1)
```

Configures the parameter 'q-RxLevMinSUL', signaled to the UE in SIB2 if the value is not OFF.

param cell_name
No help available

param power
(integer or boolean) No help available

6.3.4.10.2.498 Priority

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:RESelection:PRIority
```

class PriorityCls

Priority commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RESelection:PRIority
value: float = driver.configure.signaling.nradio.cell.reSelection.priority.
↪get(cell_name = 'abc')
```

Configures the parameter 'cellReselectionPriority', signaled to the UE in SIB2.

param cell_name
No help available

return
priority: No help available

set(*cell_name: str*, *priority: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RESelection:PRIority
driver.configure.signaling.nradio.cell.reSelection.priority.set(cell_name = 'abc
↪', priority = 1.0)
```

Configures the parameter 'cellReselectionPriority', signaled to the UE in SIB2.

param cell_name
No help available

param priority
No help available

6.3.4.10.2.499 Search

class SearchCls

Search commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.reSelection.search.clone()
```

Subgroups

6.3.4.10.2.500 Intp

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:RESelection:SEARch:INTP
```

class IntpCls

Intp commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RESelection:SEARch:INTP
value: int = driver.configure.signaling.nradio.cell.reSelection.search.intp.
↪get(cell_name = 'abc')
```

Configures the parameter 's-IntraSearchP', signaled to the UE in SIB2.

param cell_name
No help available

return
power: No help available

set(*cell_name: str*, *power: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RESelection:SEARch:INTP
driver.configure.signaling.nradio.cell.reSelection.search.intp.set(cell_name =
↪'abc', power = 1)
```

Configures the parameter 's-IntraSearchP', signaled to the UE in SIB2.

param cell_name
No help available

param power
No help available

6.3.4.10.2.501 Intq

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:RESelection:SEARch:INTQ
```

class IntqCls

Intq commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RESelection:SEARch:INTQ
value: int or bool = driver.configure.signaling.nradio.cell.reSelection.search.
↪intq.get(cell_name = 'abc')
```

Configures the parameter 's-IntraSearchQ', signaled to the UE in SIB2 if the value is not OFF.

param cell_name
No help available

return
power: (integer or boolean) No help available

set(*cell_name: str*, *power: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RESelection:SEARch:INTQ
driver.configure.signaling.nradio.cell.reSelection.search.intq.set(cell_name =
↪'abc', power = 1)
```

Configures the parameter 's-IntraSearchQ', signaled to the UE in SIB2 if the value is not OFF.

param cell_name
No help available

param power
(integer or boolean) No help available

6.3.4.10.2.502 Ninp

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:RESelection:SEARch:NINP
```

class NinpCls

Ninp commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RESelection:SEARch:NINP
value: int or bool = driver.configure.signaling.nradio.cell.reSelection.search.
↪ninp.get(cell_name = 'abc')
```

Configures the parameter 's-NonIntraSearchP', signaled to the UE in SIB2 if the value is not OFF.

param cell_name
No help available

return
power: (integer or boolean) No help available

set(*cell_name: str*, *power: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RESelection:SEARch:NINP
driver.configure.signaling.nradio.cell.reSelection.search.ninp.set(cell_name =
↪'abc', power = 1)
```

Configures the parameter 's-NonIntraSearchP', signaled to the UE in SIB2 if the value is not OFF.

param cell_name
No help available

param power
(integer or boolean) No help available

6.3.4.10.2.503 Ninq

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:RESelection:SEARch:NINQ
```

class NinqCls

Ninq commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RESelection:SEARch:NINQ
value: int or bool = driver.configure.signaling.nradio.cell.reSelection.search.
↪ninq.get(cell_name = 'abc')
```

Configures the parameter 's-NonIntraSearchQ', signaled to the UE in SIB2 if the value is not OFF.

param cell_name
No help available

return
power: (integer or boolean) No help available

set(*cell_name: str*, *power: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RESelection:SEARch:NINQ
driver.configure.signaling.nradio.cell.reSelection.search.ninq.set(cell_name =
↪'abc', power = 1)
```

Configures the parameter 's-NonIntraSearchQ', signaled to the UE in SIB2 if the value is not OFF.

param cell_name
No help available

param power
(integer or boolean) No help available

6.3.4.10.2.504 Thresholds

class ThresholdsCls

Thresholds commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.reSelection.thresholds.clone()
```

Subgroups

6.3.4.10.2.505 Lowp

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:RESelection:THResholds:LOWP
```

class LowpCls

Lowp commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RESelection:THResholds:LOWP
value: int = driver.configure.signaling.nradio.cell.reSelection.thresholds.lowp.
↪get(cell_name = 'abc')
```

Configures the parameter 'threshServingLowP', signaled to the UE in SIB2.

param cell_name
No help available

return
power: No help available

set(*cell_name: str*, *power: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RESelection:THResholds:LOWP
driver.configure.signaling.nradio.cell.reSelection.thresholds.lowp.set(cell_
↪name = 'abc', power = 1)
```

Configures the parameter 'threshServingLowP', signaled to the UE in SIB2.

param cell_name
No help available

param power
No help available

6.3.4.10.2.506 Lowq

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:RESelection:THResholds:LOWQ
```

class LowqCls

Lowq commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RESelection:THResholds:LOWQ
value: int or bool = driver.configure.signaling.nradio.cell.reSelection.
↪thresholds.lowq.get(cell_name = 'abc')
```

Configures the parameter 'threshServingLowQ', signaled to the UE in SIB2 if the value is not OFF.

param cell_name
No help available

return
power: (integer or boolean) No help available

set(*cell_name: str*, *power: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RESelection:THResholds:LOWQ
driver.configure.signaling.nradio.cell.reSelection.thresholds.lowq.set(cell_
↪name = 'abc', power = 1)
```

Configures the parameter 'threshServingLowQ', signaled to the UE in SIB2 if the value is not OFF.

param cell_name
No help available

param power
(integer or boolean) No help available

6.3.4.10.2.507 Timer

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:RESelection:TIMer
```

class TimerCls

Timer commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RESelection:TIMer
value: int = driver.configure.signaling.nradio.cell.reSelection.timer.get(cell_
↪name = 'abc')
```

Configures the parameter 't-ReselectionNR', signaled to the UE in SIB2.

param cell_name
No help available

return
time: No help available

set(*cell_name: str*, *time: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RESelection:TIMer
driver.configure.signaling.nradio.cell.reSelection.timer.set(cell_name = 'abc',␣
↪time = 1)
```

Configures the parameter 't-ReselectionNR', signaled to the UE in SIB2.

param cell_name
No help available

param time
No help available

6.3.4.10.2.508 RfSettings

class RfSettingsCls

RfSettings commands group definition. 29 total commands, 10 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.rfSettings.clone()
```

Subgroups

6.3.4.10.2.509 Apoint

class ApointCls

Apoint commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.rfSettings.apoint.clone()
```

Subgroups

6.3.4.10.2.510 Arfcn

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:APOint:ARFCn
```

class ArfcnCls

Arfcn commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:APOint:ARFCn
value: int = driver.configure.signaling.nradio.cell.rfSettings.apoint.arfcn.
↪get(cell_name = 'abc')
```

No command help available

param cell_name
No help available

return
number: No help available

set(*cell_name: str*, *number: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:APOint:ARFCn
driver.configure.signaling.nradio.cell.rfSettings.apoint.arfcn.set(cell_name =
↪'abc', number = 1)
```

No command help available

param cell_name
No help available

param number
No help available

6.3.4.10.2.511 Frequency

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:APOint:FREQuency
```

class FrequencyCls

Frequency commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:APOint:FREQuency
value: float = driver.configure.signaling.nradio.cell.rfSettings.apoint.
↪frequency.get(cell_name = 'abc')
```

No command help available

param cell_name
No help available

return
frequency: No help available

set(*cell_name: str*, *frequency: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:APOint:FREQuency
driver.configure.signaling.nradio.cell.rfSettings.apoint.frequency.set(cell_
↪name = 'abc', frequency = 1.0)
```

No command help available

param cell_name
No help available

param frequency
No help available

6.3.4.10.2.512 Location

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:APOint:LOCation
```

class LocationCls

Location commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → Location

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:APOint:LOCation
value: enums.Location = driver.configure.signaling.nradio.cell.rfSettings.
↪apoint.location.get(cell_name = 'abc')
```

No command help available

param cell_name
No help available

return
location: No help available

set(*cell_name: str*, *location: Location*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:APOint:LOCation
driver.configure.signaling.nradio.cell.rfSettings.apoint.location.set(cell_name␣
↪= 'abc', location = enums.Location.HIGH)
```

No command help available

param cell_name
No help available

param location
No help available

6.3.4.10.2.513 AsEmission

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:ASEMission
```

class AsEmissionCls

AsEmission commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:ASEMission
value: int or bool = driver.configure.signaling.nradio.cell.rfSettings.
↪asEmission.get(cell_name = 'abc')
```

Sets the parameter 'AdditionalSpectrumEmission', signaled to the UE if the value is not OFF.

param cell_name
No help available

return
as_emission: (integer or boolean) No help available

set(*cell_name: str*, *as_emission: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:ASEMission
driver.configure.signaling.nradio.cell.rfSettings.asEmission.set(cell_name =
↪'abc', as_emission = 1)
```

Sets the parameter 'AdditionalSpectrumEmission', signaled to the UE if the value is not OFF.

param cell_name
No help available

param as_emission
(integer or boolean) No help available

6.3.4.10.2.514 Combined

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:COMBined
```

class CombinedCls

Combined commands group definition. 4 total commands, 2 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Duplex_Mode: enums.DuplexModeB: No parameter help available
- Fbi: int: Frequency band indicator
- Dl_Bw: enums.DlUlBandwidth: DL carrier bandwidth in MHz
- Dl_Off_To_Carrier: int: DL offset to carrier
- Dl_Point_Aarfcn: int: DL channel number (ARFCN) of point A
- Ul_Bw: enums.DlUlBandwidth: UL carrier bandwidth in MHz (ignored for TDD/SDL)
- Ul_Off_To_Carrier: int: UL offset to carrier (ignored for TDD/SDL)
- Ul_Point_Aarfcn: int: UL channel number (ARFCN) of point A (ignored for TDD/SDL)
- Control_Zero: int: Common control resource set (CORESET) number 0
- Kssb: int: Number of SC between the SSB and the overall RB grid (kSSB) .
- Offset_Point_A: int: Parameter 'offsetToPointA' of the SIB (number of RB)
- Scs: int: Subcarrier spacing

class SetStruct

Structure for setting input parameters. Contains optional setting parameters. Fields:

- Cell_Name: str: No parameter help available
- Duplex_Mode: enums.DuplexModeB: No parameter help available
- Fbi: int: Optional setting parameter. Frequency band indicator

- Dl_Bw: enums.DlUlBandwidth: Optional setting parameter. DL carrier bandwidth in MHz
- Dl_Off_To_Carrier: int: Optional setting parameter. DL offset to carrier
- Dl_Point_Aarfcn: int: Optional setting parameter. DL channel number (ARFCN) of point A
- Ul_Bw: enums.DlUlBandwidth: Optional setting parameter. UL carrier bandwidth in MHz (ignored for TDD/SDL)
- Ul_Off_To_Carrier: int: Optional setting parameter. UL offset to carrier (ignored for TDD/SDL)
- Ul_Point_Aarfcn: int: Optional setting parameter. UL channel number (ARFCN) of point A (ignored for TDD/SDL)
- Control_Zero: int: Optional setting parameter. Common control resource set (CORESET) number 0
- Kssb: int: Optional setting parameter. Number of SC between the SSB and the overall RB grid (kSSB) .
- Offset_Point_A: int: Optional setting parameter. Parameter 'offsetToPointA' of the SIB (number of RB)
- Scs: int: Optional setting parameter. Subcarrier spacing

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:COMBined
value: GetStruct = driver.configure.signaling.nradio.cell.rfSettings.combined.
↪get(cell_name = 'abc')
```

Modifies several frequency settings simultaneously, for example, to change the frequency for an established connection, without losing the connection.

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*structure: SetStruct*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:COMBined
structure = driver.configure.signaling.nradio.cell.rfSettings.combined.
↪SetStruct()
structure.Cell_Name: str = 'abc'
structure.Duplex_Mode: enums.DuplexModeB = enums.DuplexModeB.FDD
structure.Fbi: int = 1
structure.Dl_Bw: enums.DlUlBandwidth = enums.DlUlBandwidth.B005
structure.Dl_Off_To_Carrier: int = 1
structure.Dl_Point_Aarfcn: int = 1
structure.Ul_Bw: enums.DlUlBandwidth = enums.DlUlBandwidth.B005
structure.Ul_Off_To_Carrier: int = 1
structure.Ul_Point_Aarfcn: int = 1
structure.Control_Zero: int = 1
structure.Kssb: int = 1
structure.Offset_Point_A: int = 1
structure.Scs: int = 1
driver.configure.signaling.nradio.cell.rfSettings.combined.set(structure)
```

Modifies several frequency settings simultaneously, for example, to change the frequency for an established connection, without losing the connection.

param structure
for set value, see the help for SetStruct structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.rfSettings.combined.clone()
```

Subgroups

6.3.4.10.2.515 Cfrequency

class CfrequencyCls

Cfrequency commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.rfSettings.combined.cfrequency.clone()
```

Subgroups

6.3.4.10.2.516 Arfcn

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:COMBined:CFRequency:ARFCn
```

class ArfcnCls

Arfcn commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Duplex_Mode: enums.DuplexModeB: No parameter help available
- Fbi: int: Frequency band indicator
- Dl_Bw: enums.DlUlBandwidth: DL carrier bandwidth in MHz
- Dl_Arfcn: int: DL channel number for carrier center frequency
- Ul_Bw: enums.DlUlBandwidth: UL carrier bandwidth in MHz (ignored for TDD/SDL)
- Ul_Arfcn: int: UL channel number for carrier center frequency (ignored for TDD/SDL)
- Scs: int: No parameter help available

class SetStruct

Structure for setting input parameters. Contains optional setting parameters. Fields:

- Cell_Name: str: No parameter help available

- Duplex_Mode: enums.DuplexModeB: No parameter help available
- Fbi: int: Optional setting parameter. Frequency band indicator
- Dl_Bw: enums.DlUlBandwidth: Optional setting parameter. DL carrier bandwidth in MHz
- Dl_Arfcn: int: Optional setting parameter. DL channel number for carrier center frequency
- Ul_Bw: enums.DlUlBandwidth: Optional setting parameter. UL carrier bandwidth in MHz (ignored for TDD/SDL)
- Ul_Arfcn: int: Optional setting parameter. UL channel number for carrier center frequency (ignored for TDD/SDL)
- Scs: int: No parameter help available

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:COMBined:CFRequency:ARFCn
value: GetStruct = driver.configure.signaling.nradio.cell.rfSettings.combined.
↪cfrequency.arfcn.get(cell_name = 'abc')
```

Modifies several frequency settings simultaneously, for example, to change the frequency for an established connection, without losing the connection.

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*structure: SetStruct*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:COMBined:CFRequency:ARFCn
structure = driver.configure.signaling.nradio.cell.rfSettings.combined.
↪cfrequency.arfcn.SetStruct()
structure.Cell_Name: str = 'abc'
structure.Duplex_Mode: enums.DuplexModeB = enums.DuplexModeB.FDD
structure.Fbi: int = 1
structure.Dl_Bw: enums.DlUlBandwidth = enums.DlUlBandwidth.B005
structure.Dl_Arfcn: int = 1
structure.Ul_Bw: enums.DlUlBandwidth = enums.DlUlBandwidth.B005
structure.Ul_Arfcn: int = 1
structure.Scs: int = 1
driver.configure.signaling.nradio.cell.rfSettings.combined.cfrequency.arfcn.
↪set(structure)
```

Modifies several frequency settings simultaneously, for example, to change the frequency for an established connection, without losing the connection.

param structure
for set value, see the help for SetStruct structure arguments.

6.3.4.10.2.517 Frequency

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:COMBined:CFRequency:FREQuency
```

class FrequencyCls

Frequency commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Duplex_Mode: enums.DuplexModeB: No parameter help available
- Fbi: int: Frequency band indicator
- Dl_Bw: enums.DlUlBandwidth: DL carrier bandwidth in MHz
- Dl_Frequency: float: DL carrier center frequency
- Ul_Bw: enums.DlUlBandwidth: UL carrier bandwidth in MHz (ignored for TDD/SDL)
- Ul_Frequency: float: UL carrier center frequency (ignored for TDD/SDL)
- Scs: int: No parameter help available

class SetStruct

Structure for setting input parameters. Contains optional setting parameters. Fields:

- Cell_Name: str: No parameter help available
- Duplex_Mode: enums.DuplexModeB: No parameter help available
- Fbi: int: Optional setting parameter. Frequency band indicator
- Dl_Bw: enums.DlUlBandwidth: Optional setting parameter. DL carrier bandwidth in MHz
- Dl_Frequency: float: Optional setting parameter. DL carrier center frequency
- Ul_Bw: enums.DlUlBandwidth: Optional setting parameter. UL carrier bandwidth in MHz (ignored for TDD/SDL)
- Ul_Frequency: float: Optional setting parameter. UL carrier center frequency (ignored for TDD/SDL)
- Scs: int: No parameter help available

get(*cell_name: str*) → GetStruct

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:COMBined:CFRequency:FREQuency
value: GetStruct = driver.configure.signaling.nradio.cell.rfSettings.combined.
↪cfrequency.frequency.get(cell_name = 'abc')
```

Modifies several frequency settings simultaneously, for example, to change the frequency for an established connection, without losing the connection.

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*structure: SetStruct*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:COMBined:CFRequency:FREQuency
structure = driver.configure.signaling.nradio.cell.rfSettings.combined.
↪cfrequency.frequency.SetStruct()
structure.Cell_Name: str = 'abc'
structure.Duplex_Mode: enums.DuplexModeB = enums.DuplexModeB.FDD
structure.Fbi: int = 1
structure.Dl_Bw: enums.DlUlBandwidth = enums.DlUlBandwidth.B005
structure.Dl_Frequency: float = 1.0
structure.Ul_Bw: enums.DlUlBandwidth = enums.DlUlBandwidth.B005
structure.Ul_Frequency: float = 1.0
structure.Scs: int = 1
driver.configure.signaling.nradio.cell.rfSettings.combined.cfrequency.frequency.
↪set(structure)
```

Modifies several frequency settings simultaneously, for example, to change the frequency for an established connection, without losing the connection.

param structure
for set value, see the help for SetStruct structure arguments.

6.3.4.10.2.518 Location

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:COMBined:LOCation
```

class LocationCls

Location commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Duplex_Mode: enums.DuplexModeB: No parameter help available
- Fbi: int: Frequency band indicator
- Dl_Bw: enums.DlUlBandwidth: DL carrier bandwidth in MHz
- **Dl_Location: enums.DlUlLocation: DL frequency**
  - MID, LOW, HIGH: Automatic selection of mid, low or high position in the frequency band.
  - USER: User-defined frequency, specified via a separate command.
- Ul_Bw: enums.DlUlBandwidth: UL carrier bandwidth in MHz (ignored for TDD/SDL)
- **Ul_Location: enums.DlUlLocation: UL frequency (ignored for TDD/SDL)**
  - MID, LOW, HIGH: Automatic selection of mid, low or high position in the frequency band.
  - USER: User-defined frequency, specified via a separate command.
- Scs: int: Subcarrier spacing

class SetStruct

Structure for setting input parameters. Contains optional setting parameters. Fields:

- Cell_Name: str: No parameter help available
- Duplex_Mode: enums.DuplexModeB: No parameter help available
- Fbi: int: Optional setting parameter. Frequency band indicator
- Dl_Bw: enums.DlUlBandwidth: Optional setting parameter. DL carrier bandwidth in MHz
- **Dl_Location: enums.DlUlLocation: Optional setting parameter. DL frequency**
  - MID, LOW, HIGH: Automatic selection of mid, low or high position in the frequency band.
  - USER: User-defined frequency, specified via a separate command.
- Ul_Bw: enums.DlUlBandwidth: Optional setting parameter. UL carrier bandwidth in MHz (ignored for TDD/SDL)
- **Ul_Location: enums.DlUlLocation: Optional setting parameter. UL frequency (ignored for TDD/SDL)**
  - MID, LOW, HIGH: Automatic selection of mid, low or high position in the frequency band.
  - USER: User-defined frequency, specified via a separate command.
- Scs: int: Optional setting parameter. Subcarrier spacing

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:COMBined:LOCation
value: GetStruct = driver.configure.signaling.nradio.cell.rfSettings.combined.
↪location.get(cell_name = 'abc')
```

Modifies several frequency settings simultaneously, for example, to change the frequency for an established connection, without losing the connection.

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*structure: SetStruct*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:COMBined:LOCation
structure = driver.configure.signaling.nradio.cell.rfSettings.combined.location.
↪SetStruct()
structure.Cell_Name: str = 'abc'
structure.Duplex_Mode: enums.DuplexModeB = enums.DuplexModeB.FDD
structure.Fbi: int = 1
structure.Dl_Bw: enums.DlUlBandwidth = enums.DlUlBandwidth.B005
structure.Dl_Location: enums.DlUlLocation = enums.DlUlLocation.HIGH
structure.Ul_Bw: enums.DlUlBandwidth = enums.DlUlBandwidth.B005
structure.Ul_Location: enums.DlUlLocation = enums.DlUlLocation.HIGH
structure.Scs: int = 1
driver.configure.signaling.nradio.cell.rfSettings.combined.location.
↪set(structure)
```

Modifies several frequency settings simultaneously, for example, to change the frequency for an established connection, without losing the connection.

param structure
for set value, see the help for SetStruct structure arguments.

6.3.4.10.2.519 Dmode

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:DMODe
```

class DmodeCls

Dmode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → DuplexModeB

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:DMODe
value: enums.DuplexModeB = driver.configure.signaling.nradio.cell.rfSettings.
↪dmode.get(cell_name = 'abc')
```

Selects the duplex mode.

param cell_name
No help available

return
duplex_mode: No help available

set(*cell_name: str*, *duplex_mode: DuplexModeB*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:DMODe
driver.configure.signaling.nradio.cell.rfSettings.dmode.set(cell_name = 'abc',␣
↪duplex_mode = enums.DuplexModeB.FDD)
```

Selects the duplex mode.

param cell_name
No help available

param duplex_mode
No help available

6.3.4.10.2.520 Downlink

class DownlinkCls

Downlink commands group definition. 8 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.rfSettings.downlink.clone()
```

Subgroups

6.3.4.10.2.521 Apoint

class ApointCls

Apoint commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.rfSettings.downlink.apoint.clone()
```

Subgroups

6.3.4.10.2.522 Arfcn

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:DL:APOint:ARFCn
```

class ArfcnCls

Arfcn commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:DL:APOint:ARFCn
value: int = driver.configure.signaling.nradio.cell.rfSettings.downlink.apoint.
↪arfcn.get(cell_name = 'abc')
```

Sets the user-defined channel number of point A, for the downlink.

param cell_name
No help available

return
number: No help available

set(*cell_name: str*, *number: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:DL:APOint:ARFCn
driver.configure.signaling.nradio.cell.rfSettings.downlink.apoint.arfcn.
↪set(cell_name = 'abc', number = 1)
```

Sets the user-defined channel number of point A, for the downlink.

param cell_name
No help available

param number
No help available

6.3.4.10.2.523 Frequency

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:DL:APOint:FREQuency
```

class FrequencyCls

Frequency commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:DL:APOint:FREQuency
value: float = driver.configure.signaling.nradio.cell.rfSettings.downlink.
↪apoint.frequency.get(cell_name = 'abc')
```

Sets the user-defined frequency of point A for the downlink.

param cell_name
No help available

return
frequency: No help available

set(*cell_name: str*, *frequency: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:DL:APOint:FREQuency
driver.configure.signaling.nradio.cell.rfSettings.downlink.apoint.frequency.
↪set(cell_name = 'abc', frequency = 1.0)
```

Sets the user-defined frequency of point A for the downlink.

param cell_name
No help available

param frequency
No help available

6.3.4.10.2.524 Location

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:DL:APOint:LOCation
```

class LocationCls

Location commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → Location

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:DL:APOint:LOCation
value: enums.Location = driver.configure.signaling.nradio.cell.rfSettings.
↪downlink.apoint.location.get(cell_name = 'abc')
```

Selects a method for defining the downlink frequency.

param cell_name
No help available

return
location: - MID, LOW, HIGH: Automatic selection of mid, low or high position in the frequency band. - USER: User-defined frequency, specified via a separate command.

set(*cell_name: str*, *location: Location*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:DL:APOint:LOCation
driver.configure.signaling.nradio.cell.rfSettings.downlink.apoint.location.
↪set(cell_name = 'abc', location = enums.Location.HIGH)
```

Selects a method for defining the downlink frequency.

param cell_name
No help available

param location

- MID, LOW, HIGH: Automatic selection of mid, low or high position in the frequency band.
- USER: User-defined frequency, specified via a separate command.

6.3.4.10.2.525 Bandwidth

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:DL:BWIDth
```

class BandwidthCls

Bandwidth commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:DL:BWIDth
value: float = driver.configure.signaling.nradio.cell.rfSettings.downlink.
↪bandwidth.get(cell_name = 'abc')
```

Selects the carrier bandwidth for the downlink.

param cell_name
No help available

return
bandwidth: Numeric value in Hz or enumerated value in MHz.

set(*cell_name: str*, *bandwidth: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:DL:BWIDth
driver.configure.signaling.nradio.cell.rfSettings.downlink.bandwidth.set(cell_
↪name = 'abc', bandwidth = 1.0)
```

Selects the carrier bandwidth for the downlink.

param cell_name
No help available

param bandwidth
Numeric value in Hz or enumerated value in MHz.

6.3.4.10.2.526 Cfrequency

class CfrequencyCls

Cfrequency commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.rfSettings.downlink.cfrequency.clone()
```

Subgroups

6.3.4.10.2.527 Arfcn

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:DL:CFRequency:ARFCn
```

class ArfcnCls

Arfcn commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:DL:CFRequency:ARFCn
value: int = driver.configure.signaling.nradio.cell.rfSettings.downlink.
↪cfrequency.arfcn.get(cell_name = 'abc')
```

Configures the ARFCN for the DL carrier center frequency

param cell_name
No help available

return
number: No help available

set(*cell_name: str*, *number: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:DL:CFRequency:ARFCn
driver.configure.signaling.nradio.cell.rfSettings.downlink.cfrequency.arfcn.
↪set(cell_name = 'abc', number = 1)
```

Configures the ARFCN for the DL carrier center frequency

param cell_name
No help available

param number
No help available

6.3.4.10.2.528 Frequency

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:DL:CFRequency:FREQuency
```

class FrequencyCls

Frequency commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:DL:CFRequency:FREQuency
value: float = driver.configure.signaling.nradio.cell.rfSettings.downlink.
↪cfrequency.frequency.get(cell_name = 'abc')
```

Configures the DL carrier center frequency.

param cell_name
No help available

return
frequency: No help available

set(*cell_name: str*, *frequency: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:DL:CFRequency:FREQuency
driver.configure.signaling.nradio.cell.rfSettings.downlink.cfrequency.frequency.
↪set(cell_name = 'abc', frequency = 1.0)
```

Configures the DL carrier center frequency.

param cell_name
No help available

param frequency
No help available

6.3.4.10.2.529 Ibwp

class IbwpCls

Ibwp commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.rfSettings.downlink.ibwp.clone()
```

Subgroups

6.3.4.10.2.530 Lobw

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:DL:IBWP:LOBW
```

class LobwCls

Lobw commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:DL:IBWP:LOBW
value: int = driver.configure.signaling.nradio.cell.rfSettings.downlink.ibwp.
↪lobw.get(cell_name = 'abc')
```

No command help available

param cell_name
No help available

return
lobw: No help available

set(*cell_name: str*, *lobw: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:DL:IBWP:LOBW
driver.configure.signaling.nradio.cell.rfSettings.downlink.ibwp.lobw.set(cell_
↪name = 'abc', lobw = 1)
```

No command help available

param cell_name
No help available

param lobw
No help available

6.3.4.10.2.531 Ocarrier

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:DL:OCARrier
```

class OcarrierCls

Ocarrier commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:DL:OCARrier
value: int = driver.configure.signaling.nradio.cell.rfSettings.downlink.
↪ocarrier.get(cell_name = 'abc')
```

Defines the offset to carrier for the downlink.

param cell_name
No help available

return
offset_to_carrier: No help available

set(*cell_name: str*, *offset_to_carrier: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:DL:OCARrier
driver.configure.signaling.nradio.cell.rfSettings.downlink.ocarrier.set(cell_
↪name = 'abc', offset_to_carrier = 1)
```

Defines the offset to carrier for the downlink.

param cell_name
No help available

param offset_to_carrier
No help available

6.3.4.10.2.532 FbIndicator

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:FBINdicator
```

class FbIndicatorCls

FbIndicator commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:FBINdicator
value: int = driver.configure.signaling.nradio.cell.rfSettings.fbIndicator.
↪get(cell_name = 'abc')
```

Defines the frequency band.

param cell_name
No help available

return
fbi: No help available

set(*cell_name: str*, *fbi: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:FBINdicator
driver.configure.signaling.nradio.cell.rfSettings.fbIndicator.set(cell_name =
↪'abc', fbi = 1)
```

Defines the frequency band.

param cell_name
No help available

param fbi
No help available

6.3.4.10.2.533 Frange

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:FRANge
```

class FrangeCls

Frange commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → FrequencyRange

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:FRANge
value: enums.FrequencyRange = driver.configure.signaling.nradio.cell.rfSettings.
↪frange.get(cell_name = 'abc')
```

Selects the frequency range, FR1 or FR2.

param cell_name
No help available

return
frequency_range: No help available

set(*cell_name: str*, *frequency_range: FrequencyRange*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:FRANge
driver.configure.signaling.nradio.cell.rfSettings.frange.set(cell_name = 'abc',␣
↪frequency_range = enums.FrequencyRange.FR1)
```

Selects the frequency range, FR1 or FR2.

param cell_name
No help available

param frequency_range
No help available

6.3.4.10.2.534 RbMax

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:RBMax
```

class RbMaxCls

RbMax commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:RBMax
value: bool = driver.configure.signaling.nradio.cell.rfSettings.rbMax.get(cell_
↪name = 'abc')
```

It enables the automatic configuration of a full scheduled RB allocation when the carrier bandwidth is increased.

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:RBMax
driver.configure.signaling.nradio.cell.rfSettings.rbMax.set(cell_name = 'abc',␣
↪enable = False)
```

It enables the automatic configuration of a full scheduled RB allocation when the carrier bandwidth is increased.

param cell_name
No help available

param enable
No help available

6.3.4.10.2.535 Sspacing

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:SSPacing
```

class SspacingCls

Sspacing commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:SSPacing
value: int = driver.configure.signaling.nradio.cell.rfSettings.sspacing.
↪get(cell_name = 'abc')
```

Selects the common subcarrier spacing of the cell.

param cell_name
No help available

return
spacing: No help available

set(*cell_name: str*, *spacing: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:SSPacing
driver.configure.signaling.nradio.cell.rfSettings.sspacing.set(cell_name = 'abc
↪', spacing = 1)
```

Selects the common subcarrier spacing of the cell.

param cell_name
No help available

param spacing
No help available

6.3.4.10.2.536 Uplink

class UplinkCls

Uplink commands group definition. 8 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.rfSettings.uplink.clone()
```

Subgroups

6.3.4.10.2.537 Apoint

class ApointCls

Apoint commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.rfSettings.uplink.apoint.clone()
```

Subgroups

6.3.4.10.2.538 Arfcn

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:UL:APOint:ARFCn
```

class ArfcnCls

Arfcn commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:UL:APOint:ARFCn
value: int = driver.configure.signaling.nradio.cell.rfSettings.uplink.apoint.
↪arfcn.get(cell_name = 'abc')
```

Sets the user-defined channel number of point A, for the uplink.

param cell_name
No help available

return
number: No help available

set(*cell_name: str*, *number: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:UL:APOint:ARFCn
driver.configure.signaling.nradio.cell.rfSettings.uplink.apoint.arfcn.set(cell_
↪name = 'abc', number = 1)
```

Sets the user-defined channel number of point A, for the uplink.

param cell_name
No help available

param number
No help available

6.3.4.10.2.539 Frequency

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:UL:APOint:FREQuency
```

class FrequencyCls

Frequency commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:UL:APOint:FREQuency
value: float = driver.configure.signaling.nradio.cell.rfSettings.uplink.apoint.
↪frequency.get(cell_name = 'abc')
```

Sets the user-defined frequency of point A, for the uplink.

param cell_name
No help available

return
frequency: No help available

set(*cell_name: str*, *frequency: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:UL:APOint:FREQuency
driver.configure.signaling.nradio.cell.rfSettings.uplink.apoint.frequency.
↪set(cell_name = 'abc', frequency = 1.0)
```

Sets the user-defined frequency of point A, for the uplink.

param cell_name
No help available

param frequency
No help available

6.3.4.10.2.540 Location

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:UL:APOint:LOCation
```

class LocationCls

Location commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → Location

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:UL:APOint:LOCation
value: enums.Location = driver.configure.signaling.nradio.cell.rfSettings.
↪uplink.apoint.location.get(cell_name = 'abc')
```

Selects a method for defining the uplink frequency.

param cell_name
No help available

return
location: - MID, LOW, HIGH: Automatic selection of mid, low or high position in the frequency band. - USER: User-defined frequency, specified via a separate command.

set(*cell_name: str*, *location: Location*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:UL:APOint:LOCation
driver.configure.signaling.nradio.cell.rfSettings.uplink.apoint.location.
↪set(cell_name = 'abc', location = enums.Location.HIGH)
```

Selects a method for defining the uplink frequency.

param cell_name
No help available

param location

- MID, LOW, HIGH: Automatic selection of mid, low or high position in the frequency band.
- USER: User-defined frequency, specified via a separate command.

6.3.4.10.2.541 Bandwidth

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:UL:BWIDth
```

class BandwidthCls

Bandwidth commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:UL:BWIDth
value: float = driver.configure.signaling.nradio.cell.rfSettings.uplink.
↪bandwidth.get(cell_name = 'abc')
```

Selects the carrier bandwidth for the uplink.

param cell_name
No help available

return
bandwidth: Numeric value in Hz or enumerated value in MHz.

set(*cell_name: str*, *bandwidth: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:UL:BWIDth
driver.configure.signaling.nradio.cell.rfSettings.uplink.bandwidth.set(cell_
↪name = 'abc', bandwidth = 1.0)
```

Selects the carrier bandwidth for the uplink.

param cell_name
No help available

param bandwidth
Numeric value in Hz or enumerated value in MHz.

6.3.4.10.2.542 Cfrequency

class CfrequencyCls

Cfrequency commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.rfSettings.uplink.cfrequency.clone()
```

Subgroups

6.3.4.10.2.543 Arfcn

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:UL:CFRequency:ARFCn
```

class ArfcnCls

Arfcn commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:UL:CFRequency:ARFCn
value: int = driver.configure.signaling.nradio.cell.rfSettings.uplink.
↪cfrequency.arfcn.get(cell_name = 'abc')
```

Configures the ARFCN for the UL carrier center frequency.

param cell_name
No help available

return
number: No help available

set(*cell_name: str*, *number: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:UL:CFRequency:ARFCn
driver.configure.signaling.nradio.cell.rfSettings.uplink.cfrequency.arfcn.
↪set(cell_name = 'abc', number = 1)
```

Configures the ARFCN for the UL carrier center frequency.

param cell_name
No help available

param number
No help available

6.3.4.10.2.544 Frequency

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:UL:CFRequency:FREQuency
```

class FrequencyCls

Frequency commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:UL:CFRequency:FREQuency
value: float = driver.configure.signaling.nradio.cell.rfSettings.uplink.
↪cfrequency.frequency.get(cell_name = 'abc')
```

Configures the UL carrier center frequency.

param cell_name
No help available

return
frequency: No help available

set(*cell_name: str*, *frequency: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:UL:CFRequency:FREQuency
driver.configure.signaling.nradio.cell.rfSettings.uplink.cfrequency.frequency.
↪set(cell_name = 'abc', frequency = 1.0)
```

Configures the UL carrier center frequency.

param cell_name
No help available

param frequency
No help available

6.3.4.10.2.545 Ibwp

class IbwpCls

Ibwp commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.rfSettings.uplink.ibwp.clone()
```

Subgroups

6.3.4.10.2.546 Lobw

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:UL:IBWP:LOBW
```

class LobwCls

Lobw commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:UL:IBWP:LOBW
value: int = driver.configure.signaling.nradio.cell.rfSettings.uplink.ibwp.lobw.
↪get(cell_name = 'abc')
```

No command help available

param cell_name
No help available

return
lobw: No help available

set(*cell_name: str*, *lobw: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:UL:IBWP:LOBW
driver.configure.signaling.nradio.cell.rfSettings.uplink.ibwp.lobw.set(cell_
↪name = 'abc', lobw = 1)
```

No command help available

param cell_name
No help available

param lobw
No help available

6.3.4.10.2.547 Ocarrier

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:UL:OCARrier
```

class OcarrierCls

Ocarrier commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:UL:OCARrier
value: int = driver.configure.signaling.nradio.cell.rfSettings.uplink.ocarrier.
↪get(cell_name = 'abc')
```

Defines the offset to carrier for the uplink.

param cell_name
No help available

return
offset_to_carrier: No help available

set(*cell_name: str*, *offset_to_carrier: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:RFSettings:UL:OCARrier
driver.configure.signaling.nradio.cell.rfSettings.uplink.ocarrier.set(cell_name␣
↪= 'abc', offset_to_carrier = 1)
```

Defines the offset to carrier for the uplink.

param cell_name
No help available

param offset_to_carrier
No help available

6.3.4.10.2.548 Srs

class SrsCls

Srs commands group definition. 24 total commands, 7 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.srs.clone()
```

Subgroups

6.3.4.10.2.549 Aswitching

class AswitchingCls

Aswitching commands group definition. 9 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.srs.aswitching.clone()
```

Subgroups

6.3.4.10.2.550 Enable

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:SRS:ASWitching:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:ASWitching:ENABle
value: bool = driver.configure.signaling.nradio.cell.srs.aswitching.enable.
↪get(cell_name = 'abc')
```

Enables or disables SRS antenna switching, for the initial BWP.

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:ASWitching:ENABle
driver.configure.signaling.nradio.cell.srs.aswitching.enable.set(cell_name =
↪'abc', enable = False)
```

Enables or disables SRS antenna switching, for the initial BWP.

param cell_name
No help available

param enable
No help available

6.3.4.10.2.551 Power

class PowerCls

Power commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.srs.aswitching.power.clone()
```

Subgroups

6.3.4.10.2.552 Alpha

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:SRS:ASWitching:POWer:ALPHa
```

class AlphaCls

Alpha commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → Alpha

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:ASWitching:POWer:ALPHa
value: enums.Alpha = driver.configure.signaling.nradio.cell.srs.aswitching.
↪power.alpha.get(cell_name = 'abc')
```

Sets the SRS power control parameter ‘alpha’ for SRS antenna switching, for the initial BWP.

param cell_name
No help available

return
alpha: Axy means x.y.

set(*cell_name: str*, *alpha: Alpha*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:ASWitching:POWer:ALPHa
driver.configure.signaling.nradio.cell.srs.aswitching.power.alpha.set(cell_name
↪= 'abc', alpha = enums.Alpha.A00)
```

Sets the SRS power control parameter ‘alpha’ for SRS antenna switching, for the initial BWP.

param cell_name
No help available

param alpha
Axy means x.y.

6.3.4.10.2.553 Pzero

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:SRS:ASWitching:POWer:PZERo
```

class PzeroCls

Pzero commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:ASWitching:POWer:PZERo
value: int = driver.configure.signaling.nradio.cell.srs.aswitching.power.pzero.
↪get(cell_name = 'abc')
```

Sets the SRS power control parameter ‘p0’ for SRS antenna switching, for the initial BWP.

param cell_name
No help available

return
p_0: No help available

set(*cell_name: str*, *p_0: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:ASWitching:POWer:PZERo
driver.configure.signaling.nradio.cell.srs.aswitching.power.pzero.set(cell_name
↪= 'abc', p_0 = 1)
```

Sets the SRS power control parameter ‘p0’ for SRS antenna switching, for the initial BWP.

param cell_name
No help available

param p_0
No help available

6.3.4.10.2.554 Resource

class ResourceCls

Resource commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.srs.aswitching.resource.clone()
```

Subgroups

6.3.4.10.2.555 Fhopping

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:SRS:ASWitching:RESource:FHOPping
```

class FhoppingCls

Fhopping commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Csrs: int: No parameter help available
- Bsrs: int: No parameter help available
- Bhop: int: No parameter help available

get(*cell_name: str*, *resource_no: int*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:ASWitching:RESource:FHOPping
value: GetStruct = driver.configure.signaling.nradio.cell.srs.aswitching.
↪resource.fhopping.get(cell_name = 'abc', resource_no = 1)
```

Configures the frequency hopping for SRS resource <ResourceNo> for SRS antenna switching, for the initial BWP.

param cell_name
No help available

param resource_no
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *resource_no: int*, *csrs: int*, *bsrs: int = None*, *bhop: int = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:ASWitching:RESource:FHOPping
driver.configure.signaling.nradio.cell.srs.aswitching.resource.fhopping.
↪set(cell_name = 'abc', resource_no = 1, csrs = 1, bsrs = 1, bhop = 1)
```

Configures the frequency hopping for SRS resource <ResourceNo> for SRS antenna switching, for the initial BWP.

param cell_name
No help available

param resource_no
No help available

param csrs
No help available

param bsrs
No help available

param bhop
No help available

6.3.4.10.2.556 Resource

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:SRS:ASWitching:RESource:RESource
```

`class ResourceCls`

Resource commands group definition. 1 total commands, 0 Subgroups, 1 group commands

`class GetStruct`

Response structure. Fields:

- Resource_Id: int: No parameter help available
- No_Srs_Ports: enums.AntNoPorts: Number of antenna ports for SRS transmissions
- Fd_Position: int: Frequency domain position
- Fd_Shift: int: Frequency domain shift
- Sequence_Id: int: No parameter help available

`class SetStruct`

Structure for setting input parameters. Contains optional setting parameters. Fields:

- Cell_Name: str: No parameter help available
- Resource_No: int: No parameter help available
- Resource_Id: int: No parameter help available
- No_Srs_Ports: enums.AntNoPorts: Optional setting parameter. Number of antenna ports for SRS transmissions
- Fd_Position: int: Optional setting parameter. Frequency domain position
- Fd_Shift: int: Optional setting parameter. Frequency domain shift
- Sequence_Id: int: No parameter help available

`get`(*cell_name: str*, *resource_no: int*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:ASWitching:RESource:RESource
value: GetStruct = driver.configure.signaling.nradio.cell.srs.aswitching.
↪resource.resource.get(cell_name = 'abc', resource_no = 1)
```

Configures the SRS resource <ResourceNo> for SRS antenna switching, for the initial BWP.

param cell_name
No help available

param resource_no
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*structure: SetStruct*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:ASWitching:RESource:RESource
structure = driver.configure.signaling.nradio.cell.srs.aswitching.resource.
↪resource.SetStruct()
structure.Cell_Name: str = 'abc'
structure.Resource_No: int = 1
structure.Resource_Id: int = 1
structure.No_Srs_Ports: enums.AntNoPorts = enums.AntNoPorts.P1
structure.Fd_Position: int = 1
structure.Fd_Shift: int = 1
structure.Sequence_Id: int = 1
driver.configure.signaling.nradio.cell.srs.aswitching.resource.resource.
↪set(structure)
```

Configures the SRS resource <ResourceNo> for SRS antenna switching, for the initial BWP.

param structure
for set value, see the help for SetStruct structure arguments.

6.3.4.10.2.557 Rmapping

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:SRS:ASWitching:RESource:RMAPping
```

class RmappingCls

Rmapping commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Start_Position: int: No parameter help available
- No_Symbols: enums.NoSymbolsN: No parameter help available
- Rep_Factor: enums.NoSymbolsN: Repetition factor

get(*cell_name: str*, *resource_no: int*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:ASWitching:RESource:RMAPping
value: GetStruct = driver.configure.signaling.nradio.cell.srs.aswitching.
↪resource.rmapping.get(cell_name = 'abc', resource_no = 1)
```

Configures the resource mapping for SRS transmissions, for SRS resource <ResourceNo> for SRS antenna switching, for the initial BWP.

param cell_name
No help available

param resource_no
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *resource_no: int*, *start_position: int*, *no_symbols: NoSymbolsN = None*, *rep_factor: NoSymbolsN = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:ASWitching:RESource:RMAPping
driver.configure.signaling.nradio.cell.srs.aswitching.resource.rmapping.
↪set(cell_name = 'abc', resource_no = 1, start_position = 1, no_symbols =␣
↪enums.NoSymbolsN.N1, rep_factor = enums.NoSymbolsN.N1)
```

Configures the resource mapping for SRS transmissions, for SRS resource <ResourceNo> for SRS antenna switching, for the initial BWP.

param cell_name
No help available

param resource_no
No help available

param start_position
No help available

param no_symbols
No help available

param rep_factor
Repetition factor

6.3.4.10.2.558 Rtype

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:SRS:ASWitching:RESource:RTYPe
```

class RtypeCls

Rtype commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Period: int: Periodicity of slots (SRS every nth slot)
- Offset: int: Offset as number of slots. Must be smaller than the Period.

get(*cell_name: str*, *resource_no: int*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:ASWitching:RESource:RTYPe
value: GetStruct = driver.configure.signaling.nradio.cell.srs.aswitching.
↪resource.rtype.get(cell_name = 'abc', resource_no = 1)
```

Configures the resource type for SRS resource <ResourceNo> for SRS antenna switching, for the initial BWP.

param cell_name
No help available

param resource_no
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *resource_no: int*, *period: int = None*, *offset: int = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:ASWitching:RESource:RTYPe
driver.configure.signaling.nradio.cell.srs.aswitching.resource.rtype.set(cell_
↪name = 'abc', resource_no = 1, period = 1, offset = 1)
```

Configures the resource type for SRS resource <ResourceNo> for SRS antenna switching, for the initial BWP.

param cell_name
No help available

param resource_no
No help available

param period
Periodicity of slots (SRS every nth slot)

param offset
Offset as number of slots. Must be smaller than the Period.

6.3.4.10.2.559 Tcomb

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:SRS:ASWitching:RESource:TCOMb
```

class TcombCls

Tcomb commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Ktc: enums.Ktc: No parameter help available
- Offset: int: No parameter help available
- Cyclic_Shift: int: No parameter help available

get(*cell_name: str*, *resource_no: int*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:ASWitching:RESource:TCOMb
value: GetStruct = driver.configure.signaling.nradio.cell.srs.aswitching.
↪resource.tcomb.get(cell_name = 'abc', resource_no = 1)
```

Configures the comb structure of the SRS resource <ResourceNo> for SRS antenna switching.

param cell_name
No help available

param resource_no
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *resource_no: int*, *ktc: Ktc = None*, *offset: int = None*, *cyclic_shift: int = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:ASWitching:RESource:TCOMb
driver.configure.signaling.nradio.cell.srs.aswitching.resource.tcomb.set(cell_
↪name = 'abc', resource_no = 1, ktc = enums.Ktc.N2, offset = 1, cyclic_shift =␣
↪1)
```

Configures the comb structure of the SRS resource <ResourceNo> for SRS antenna switching.

param cell_name
No help available

param resource_no
No help available

param ktc
No help available

param offset
No help available

param cyclic_shift
No help available

6.3.4.10.2.560 TypePy

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:SRS:ASWitching:TYPE
```

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → AswitchingType

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:ASWitching:TYPE
value: enums.AswitchingType = driver.configure.signaling.nradio.cell.srs.
↪aswitching.typePy.get(cell_name = 'abc')
```

Selects the antenna switching resource type for the initial BWP.

param cell_name
No help available

return
type_py: TtRr defines the number of ports t per SRS resource and the total number of ports over all SRS resources r.

set(*cell_name: str*, *type_py: AswitchingType*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:ASWitching:TYPE
driver.configure.signaling.nradio.cell.srs.aswitching.typePy.set(cell_name =
↪'abc', type_py = enums.AswitchingType.T1R1)
```

Selects the antenna switching resource type for the initial BWP.

param cell_name
No help available

param type_py
TtRr defines the number of ports t per SRS resource and the total number of ports over all SRS resources r.

6.3.4.10.2.561 CnCodebook

class CnCodebookCls

CnCodebook commands group definition. 10 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.srs.cnCodebook.clone()
```

Subgroups

6.3.4.10.2.562 Power

class PowerCls

Power commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.srs.cnCodebook.power.clone()
```

Subgroups

6.3.4.10.2.563 Alpha

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:SRS:CNCodebook:POWer:ALPHa
```

class AlphaCls

Alpha commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → Alpha

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:CNCodebook:POWer:ALPHa
value: enums.Alpha = driver.configure.signaling.nradio.cell.srs.cnCodebook.
↪power.alpha.get(cell_name = 'abc')
```

Sets the SRS power control parameter ‘alpha’ for periodic SRS, for the initial BWP.

param cell_name
No help available

return
alpha: Axy means x.y.

set(*cell_name: str*, *alpha: Alpha*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:CNCodebook:POWer:ALPHa
driver.configure.signaling.nradio.cell.srs.cnCodebook.power.alpha.set(cell_name␣
↪= 'abc', alpha = enums.Alpha.A00)
```

Sets the SRS power control parameter ‘alpha’ for periodic SRS, for the initial BWP.

param cell_name
No help available

param alpha
Axy means x.y.

6.3.4.10.2.564 Pzero

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:SRS:CNCodebook:POWer:PZERo
```

class PzeroCls

Pzero commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:CNCodebook:POWer:PZERo
value: int = driver.configure.signaling.nradio.cell.srs.cnCodebook.power.pzero.
↪get(cell_name = 'abc')
```

Sets the SRS power control parameter ‘p0’ for periodic SRS, for the initial BWP.

param cell_name
No help available

return
p_0: No help available

set(*cell_name: str*, *p_0: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:CNCodebook:POWer:PZERo
driver.configure.signaling.nradio.cell.srs.cnCodebook.power.pzero.set(cell_name
↪= 'abc', p_0 = 1)
```

Sets the SRS power control parameter 'p0' for periodic SRS, for the initial BWP.

param cell_name
No help available

param p_0
No help available

6.3.4.10.2.565 Resource

class ResourceCls

Resource commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.srs.cnCodebook.resource.clone()
```

Subgroups

6.3.4.10.2.566 Fhopping

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:SRS:CNCodebook:RESource:FHOPping
```

class FhoppingCls

Fhopping commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Csrs: int: No parameter help available
- Bsrs: int: No parameter help available
- Bhop: int: No parameter help available

get(*cell_name: str*, *resource_no: int*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:CNCodebook:RESource:FHOPping
value: GetStruct = driver.configure.signaling.nradio.cell.srs.cnCodebook.
↪resource.fhopping.get(cell_name = 'abc', resource_no = 1)
```

Configures the frequency hopping for SRS resource <ResourceNo> for periodic SRS, for the initial BWP.

param cell_name
No help available

param resource_no
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *resource_no: int*, *csrs: int*, *bsrs: int = None*, *bhop: int = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:CNCodebook:RESource:FHOPping
driver.configure.signaling.nradio.cell.srs.cnCodebook.resource.fhopping.
↪set(cell_name = 'abc', resource_no = 1, csrs = 1, bsrs = 1, bhop = 1)
```

Configures the frequency hopping for SRS resource <ResourceNo> for periodic SRS, for the initial BWP.

param cell_name
No help available

param resource_no
No help available

param csrs
No help available

param bsrs
No help available

param bhop
No help available

6.3.4.10.2.567 Resource

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:SRS:CNCodebook:RESource:RESource
```

class ResourceCls

Resource commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Resource_Id: int: No parameter help available
- No_Srs_Ports: enums.AntNoPorts: Number of antenna ports for SRS transmissions
- Fd_Position: int: Frequency domain position
- Fd_Shift: int: Frequency domain shift
- Sequence_Id: int: No parameter help available

class SetStruct

Structure for setting input parameters. Contains optional setting parameters. Fields:

- Cell_Name: str: No parameter help available
- Resource_No: int: No parameter help available
- Resource_Id: int: No parameter help available
- No_Srs_Ports: enums.AntNoPorts: Optional setting parameter. Number of antenna ports for SRS transmissions
- Fd_Position: int: Optional setting parameter. Frequency domain position
- Fd_Shift: int: Optional setting parameter. Frequency domain shift
- Sequence_Id: int: No parameter help available

get(*cell_name: str*, *resource_no: int*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:CNCodebook:RESource:RESource
value: GetStruct = driver.configure.signaling.nradio.cell.srs.cnCodebook.
↪resource.resource.get(cell_name = 'abc', resource_no = 1)
```

Configures the SRS resource <ResourceNo> for periodic SRS, for the initial BWP.

param cell_name
No help available

param resource_no
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*structure: SetStruct*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:CNCodebook:RESource:RESource
structure = driver.configure.signaling.nradio.cell.srs.cnCodebook.resource.
↪resource.SetStruct()
structure.Cell_Name: str = 'abc'
structure.Resource_No: int = 1
structure.Resource_Id: int = 1
structure.No_Srs_Ports: enums.AntNoPorts = enums.AntNoPorts.P1
structure.Fd_Position: int = 1
structure.Fd_Shift: int = 1
structure.Sequence_Id: int = 1
driver.configure.signaling.nradio.cell.srs.cnCodebook.resource.resource.
↪set(structure)
```

Configures the SRS resource <ResourceNo> for periodic SRS, for the initial BWP.

param structure
for set value, see the help for SetStruct structure arguments.

6.3.4.10.2.568 Rmapping

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:SRS:CNCodebook:RESource:RMAPping
```

class RmappingCls

Rmapping commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Start_Position: int: No parameter help available
- No_Symbols: enums.NoSymbolsN: No parameter help available
- Rep_Factor: enums.NoSymbolsN: Repetition factor

get(*cell_name: str*, *resource_no: int*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:CNCodebook:RESource:RMAPping
value: GetStruct = driver.configure.signaling.nradio.cell.srs.cnCodebook.
↪resource.rmapping.get(cell_name = 'abc', resource_no = 1)
```

Configures the resource mapping for SRS transmissions, for SRS resource <ResourceNo> for periodic SRS, for the initial BWP.

param cell_name
No help available

param resource_no
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *resource_no: int*, *start_position: int*, *no_symbols: NoSymbolsN = None*, *rep_factor: NoSymbolsN = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:CNCodebook:RESource:RMAPping
driver.configure.signaling.nradio.cell.srs.cnCodebook.resource.rmapping.
↪set(cell_name = 'abc', resource_no = 1, start_position = 1, no_symbols =␣
↪enums.NoSymbolsN.N1, rep_factor = enums.NoSymbolsN.N1)
```

Configures the resource mapping for SRS transmissions, for SRS resource <ResourceNo> for periodic SRS, for the initial BWP.

param cell_name
No help available

param resource_no
No help available

param start_position
No help available

param no_symbols
No help available

param rep_factor
Repetition factor

6.3.4.10.2.569 Rtype

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:SRS:CNCodebook:RESource:RTYPe
```

class RtypeCls

Rtype commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Td_Type: enums.TdType: APERiodic: no SRS transmissions PERiodic: SRS transmissions in every nth slot
- Period: int: Periodicity of slots (SRS every nth slot)
- Offset: int: Offset as number of slots. Must be smaller than the Period.

get(*cell_name: str*, *resource_no: int*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:CNCodebook:RESource:RTYPe
value: GetStruct = driver.configure.signaling.nradio.cell.srs.cnCodebook.
↪resource.rtype.get(cell_name = 'abc', resource_no = 1)
```

Configures the resource type for SRS resource <ResourceNo> for periodic SRS, for the initial BWP.

param cell_name
No help available

param resource_no
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *resource_no: int*, *td_type: TdType*, *period: int = None*, *offset: int = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:CNCodebook:RESource:RTYPe
driver.configure.signaling.nradio.cell.srs.cnCodebook.resource.rtype.set(cell_
↪name = 'abc', resource_no = 1, td_type = enums.TdType.APERiodic, period = 1,␣
↪offset = 1)
```

Configures the resource type for SRS resource <ResourceNo> for periodic SRS, for the initial BWP.

param cell_name
No help available

param resource_no
No help available

param td_type
APERiodic: no SRS transmissions PERiodic: SRS transmissions in every nth slot

param period
Periodicity of slots (SRS every nth slot)

param offset
Offset as number of slots. Must be smaller than the Period.

6.3.4.10.2.570 Tcomb

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:SRS:CNCodebook:RESource:TCOMb
```

class TcombCls

Tcomb commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Ktc: enums.Ktc: No parameter help available
- Offset: int: No parameter help available
- Cyclic_Shift: int: No parameter help available

get(*cell_name: str*, *resource_no: int*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:CNCodebook:RESource:TCOMb
value: GetStruct = driver.configure.signaling.nradio.cell.srs.cnCodebook.
↪resource.tcomb.get(cell_name = 'abc', resource_no = 1)
```

Configures the comb structure of the SRS resource <ResourceNo> for periodic SRS, for the initial BWP.

param cell_name
No help available

param resource_no
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *resource_no: int*, *ktc: Ktc = None*, *offset: int = None*, *cyclic_shift: int = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:CNCodebook:RESource:TCOMb
driver.configure.signaling.nradio.cell.srs.cnCodebook.resource.tcomb.set(cell_
↪name = 'abc', resource_no = 1, ktc = enums.Ktc.N2, offset = 1, cyclic_shift =␣
↪1)
```

Configures the comb structure of the SRS resource <ResourceNo> for periodic SRS, for the initial BWP.

param cell_name
No help available

param resource_no
No help available

param ktc
No help available

param offset
No help available

param cyclic_shift
No help available

6.3.4.10.2.571 Scheduler

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:SRS:CNCodebook:SCHeduler
```

class SchedulerCls

Scheduler commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Ncoherent_Tpmi: enums.NcoherentTpmi: No parameter help available
- Tpmi_Layers: enums.MaxLength: No parameter help available
- Tpmi: enums.Tpmi: No parameter help available
- Resource_Id: enums.ResourceId: No parameter help available

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:CNCodebook:SCHeduler
value: GetStruct = driver.configure.signaling.nradio.cell.srs.cnCodebook.
↪scheduler.get(cell_name = 'abc')
```

No command help available

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *ncoherent_tpmi: NcoherentTpmi*, *tpmi_layers: MaxLength = None*, *tpmi: Tpmi = None*, *resource_id: ResourceId = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:CNCodebook:SCHeduler
driver.configure.signaling.nradio.cell.srs.cnCodebook.scheduler.set(cell_name =
↪'abc', ncoherent_tpmi = enums.NcoherentTpmi.FPARtial, tpmi_layers = enums.
↪MaxLength.L1, tpmi = enums.Tpmi.T0, resource_id = enums.ResourceId.R1)
```

No command help available

param cell_name
No help available

param ncoherent_tpmi
No help available

param tpmi_layers
No help available

param tpmi
No help available

param resource_id
No help available

6.3.4.10.2.572 TdBehavior

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:SRS:CNCodebook:TDBehavior
```

class TdBehaviorCls

TdBehavior commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → TdType

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:CNCodebook:TDBehavior
value: enums.TdType = driver.configure.signaling.nradio.cell.srs.cnCodebook.
↪tdBehavior.get(cell_name = 'abc')
```

Selects the time domain behavior (‘resourceType’) of the SRS resource set and thus enables or disables periodic SRS, for the initial BWP.

param cell_name
No help available

return
td_behavior: APERiodic: no SRS transmissions PERiodic: SRS transmissions in every nth slot

set(*cell_name: str*, *td_behavior: TdType*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:CNCodebook:TDBehavior
driver.configure.signaling.nradio.cell.srs.cnCodebook.tdBehavior.set(cell_name␣
↪= 'abc', td_behavior = enums.TdType.APERiodic)
```

Selects the time domain behavior (‘resourceType’) of the SRS resource set and thus enables or disables periodic SRS, for the initial BWP.

param cell_name
No help available

param td_behavior
APERiodic: no SRS transmissions PERiodic: SRS transmissions in every nth slot

6.3.4.10.2.573 Usage

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:SRS:CNCodebook:USAGe
```

class UsageCls

Usage commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → Schema

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:CNCodebook:USAGe
value: enums.Schema = driver.configure.signaling.nradio.cell.srs.cnCodebook.
↪usage.get(cell_name = 'abc')
```

Selects the usage of the SRS resource set for periodic SRS, for the initial BWP.

param cell_name
No help available

return
schema: Codebook, non-codebook

set(*cell_name: str*, *schema: Schema*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:CNCodebook:USAGe
driver.configure.signaling.nradio.cell.srs.cnCodebook.usage.set(cell_name = 'abc
↪', schema = enums.Schema.CODebook)
```

Selects the usage of the SRS resource set for periodic SRS, for the initial BWP.

param cell_name
No help available

param schema
Codebook, non-codebook

6.3.4.10.2.574 Fhopping

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:SRS:FHOPping
```

class FhoppingCls

Fhopping commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Csrs: int: No parameter help available
- Bsrs: int: No parameter help available
- Bhop: int: No parameter help available

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:FHOPping
value: GetStruct = driver.configure.signaling.nradio.cell.srs.fhopping.get(cell_
↪name = 'abc')
```

No command help available

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *csrs: int*, *bsrs: int = None*, *bhop: int = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:FHOPping
driver.configure.signaling.nradio.cell.srs.fhopping.set(cell_name = 'abc', csrs
↪= 1, bsrs = 1, bhop = 1)
```

No command help available

param cell_name
No help available

param csrs
No help available

param bsrs
No help available

param bhop
No help available

6.3.4.10.2.575 Resource

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:SRS:RESource
```

class ResourceCls

Resource commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Resource_Id: int: No parameter help available
- No_Srs_Ports: enums.AntNoPorts: No parameter help available
- Fd_Position: int: No parameter help available
- Fd_Shift: int: No parameter help available
- Sequence_Id: int: No parameter help available

class SetStruct

Structure for setting input parameters. Contains optional setting parameters. Fields:

- Cell_Name: str: No parameter help available
- Resource_Id: int: No parameter help available
- No_Srs_Ports: enums.AntNoPorts: No parameter help available
- Fd_Position: int: No parameter help available
- Fd_Shift: int: No parameter help available
- Sequence_Id: int: No parameter help available

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:RESource
value: GetStruct = driver.configure.signaling.nradio.cell.srs.resource.get(cell_
↪name = 'abc')
```

No command help available

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*structure: SetStruct*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:RESource
structure = driver.configure.signaling.nradio.cell.srs.resource.SetStruct()
structure.Cell_Name: str = 'abc'
structure.Resource_Id: int = 1
structure.No_Srs_Ports: enums.AntNoPorts = enums.AntNoPorts.P1
structure.Fd_Position: int = 1
structure.Fd_Shift: int = 1
structure.Sequence_Id: int = 1
driver.configure.signaling.nradio.cell.srs.resource.set(structure)
```

No command help available

param structure
for set value, see the help for SetStruct structure arguments.

6.3.4.10.2.576 Rmapping

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:SRS:RMAPping
```

class RmappingCls

Rmapping commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Start_Position: int: No parameter help available
- No_Symbols: enums.NoSymbolsN: No parameter help available
- Rep_Factor: enums.NoSymbolsN: No parameter help available

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:RMAPping
value: GetStruct = driver.configure.signaling.nradio.cell.srs.rmapping.get(cell_
↪name = 'abc')
```

No command help available

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *start_position: int*, *no_symbols: NoSymbolsN = None*, *rep_factor: NoSymbolsN = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:RMAPping
driver.configure.signaling.nradio.cell.srs.rmapping.set(cell_name = 'abc',␣
↪start_position = 1, no_symbols = enums.NoSymbolsN.N1, rep_factor = enums.
↪NoSymbolsN.N1)
```

No command help available

param cell_name
No help available

param start_position
No help available

param no_symbols
No help available

param rep_factor
No help available

6.3.4.10.2.577 Rtype

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:SRS:RTYPe
```

class RtypeCls

Rtype commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Td_Type: enums.TdType: No parameter help available
- Period: int: No parameter help available
- Offset: int: No parameter help available

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:RTYPe
value: GetStruct = driver.configure.signaling.nradio.cell.srs.rtype.get(cell_
↪name = 'abc')
```

No command help available

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *td_type: TdType*, *period: int = None*, *offset: int = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:RTYPe
driver.configure.signaling.nradio.cell.srs.rtype.set(cell_name = 'abc', td_type␣
↪= enums.TdType.APERiodic, period = 1, offset = 1)
```

No command help available

param cell_name
No help available

param td_type
No help available

param period
No help available

param offset
No help available

6.3.4.10.2.578 Tcomb

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:SRS:TCOMb
```

class TcombCls

Tcomb commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Ktc: enums.Ktc: No parameter help available
- Offset: int: No parameter help available
- Cyclic_Shift: int: No parameter help available

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:TCOMb
value: GetStruct = driver.configure.signaling.nradio.cell.srs.tcomb.get(cell_
↪name = 'abc')
```

No command help available

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *ktc: Ktc = None*, *offset: int = None*, *cyclic_shift: int = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SRS:TCOMb
driver.configure.signaling.nradio.cell.srs.tcomb.set(cell_name = 'abc', ktc =␣
↪enums.Ktc.N2, offset = 1, cyclic_shift = 1)
```

No command help available

param cell_name
No help available

param ktc
No help available

param offset
No help available

param cyclic_shift
No help available

6.3.4.10.2.579 Ssb

class SsbCls

Ssb commands group definition. 11 total commands, 8 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.ssb.clone()
```

Subgroups

6.3.4.10.2.580 Afrequency

class AfrequencyCls

Afrequency commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.ssb.afrequency.clone()
```

Subgroups

6.3.4.10.2.581 Arfcn

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:SSB:AFRequency:ARFCn
```

class ArfcnCls

Arfcn commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SSB:AFRequency:ARFCn
value: int = driver.configure.signaling.nradio.cell.ssb.afrequency.arfcn.
↪get(cell_name = 'abc')
```

Queries the channel number of the SSB (‘absoluteFrequencySSB’) .

param cell_name
No help available

return
arfcn: No help available

6.3.4.10.2.582 Frequency

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:SSB:AFRequency:FREQuency
```

class FrequencyCls

Frequency commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SSB:AFRequency:FREQuency
value: float = driver.configure.signaling.nradio.cell.ssb.afrequency.frequency.
↪get(cell_name = 'abc')
```

Queries the center frequency of the SSB.

param cell_name
No help available

return
frequency: No help available

6.3.4.10.2.583 Beam

class BeamCls

Beam commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.ssb.beam.clone()
```

Subgroups

6.3.4.10.2.584 Model

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:SSB:BEAM:MODel
```

class ModelCls

Model commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Position: List[int]: Position index (0 to n) in the 'ssb-PositionsInBurst' bitmap.
- Enable: List[bool]: Use the position for SSB transmission (ON) or not (OFF) .
- Aoa: List[enums.Aoa]: Angle of arrival for the position. CONDucted: conducted test setup AOA1: Over-the-air test setup, first angle of arrival used AOA2: Over-the-air test setup, second angle of arrival used
- Phase: List[float]: Phase for the position.
- Attenuation: List[float]: Power difference for the position.

class SetStruct

Structure for setting input parameters. Contains optional setting parameters. Fields:

- Cell_Name: str: No parameter help available
- Position: List[int]: Position index (0 to n) in the 'ssb-PositionsInBurst' bitmap.
- Enable: List[bool]: Use the position for SSB transmission (ON) or not (OFF) .
- Aoa: List[enums.Aoa]: Optional setting parameter. Angle of arrival for the position. CONDucted: conducted test setup AOA1: Over-the-air test setup, first angle of arrival used AOA2: Over-the-air test setup, second angle of arrival used
- Phase: List[float]: Optional setting parameter. Phase for the position.
- Attenuation: List[float]: Optional setting parameter. Power difference for the position.

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SSB:BEAM:MODel
value: GetStruct = driver.configure.signaling.nradio.cell.ssb.beam.model.
↪get(cell_name = 'abc')
```

Configures the SS-block positions in an SSB burst and the beam properties for each position. Configuration is only possible for the mode UDEFined, selected via [CONFigure:]SIGNaling:NRADio:CELL:SSB:BEAM:PIBurst. For other modes, you can only query the automatically configured settings. You can configure several positions via one command: <CellName>, {<Position>, <Enable>, <AoA>, <Phase>, <Attenuation>}pos a, {...}pos b, ...

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*structure: SetStruct*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SSB:BEAM:MODel
structure = driver.configure.signaling.nradio.cell.ssb.beam.model.SetStruct()
structure.Cell_Name: str = 'abc'
structure.Position: List[int] = [1, 2, 3]
structure.Enable: List[bool] = [True, False, True]
structure.Aoa: List[enums.Aoa] = [Aoa.AOA1, Aoa.CONDucted]
structure.Phase: List[float] = [1.1, 2.2, 3.3]
structure.Attenuation: List[float] = [1.1, 2.2, 3.3]
driver.configure.signaling.nradio.cell.ssb.beam.model.set(structure)
```

Configures the SS-block positions in an SSB burst and the beam properties for each position. Configuration is only possible for the mode UDEFined, selected via [CONFigure:]SIGNaling:NRADio:CELL:SSB:BEAM:PIBurst. For other modes, you can only query the automatically configured settings. You can configure several positions via one command: <CellName>, {<Position>, <Enable>, <AoA>, <Phase>, <Attenuation>}pos a, {...}pos b, ...

param structure
for set value, see the help for SetStruct structure arguments.

6.3.4.10.2.585 PiBurst

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:SSB:BEAM:PIBurst
```

class PiBurstCls

PiBurst commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → BeamConfigMode

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SSB:BEAM:PIBurst
value: enums.BeamConfigMode = driver.configure.signaling.nradio.cell.ssb.beam.
↪piBurst.get(cell_name = 'abc')
```

Selects a mode for configuration of the SS-block positions.

param cell_name
No help available

return
beam_config_mode: AUTO: Use only position 1 (bitmap 0100...) . ALL: Use all possible positions (bitmap 1111...) . UDEFined: See [CONFigure:]SIGNaling:NRADio:CELL:SSB:BEAM:MODel.

set(*cell_name: str*, *beam_config_mode: BeamConfigMode*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SSB:BEAM:PIBurst
driver.configure.signaling.nradio.cell.ssb.beam.piBurst.set(cell_name = 'abc',␣
↪beam_config_mode = enums.BeamConfigMode.ALL)
```

Selects a mode for configuration of the SS-block positions.

param cell_name
No help available

param beam_config_mode
AUTO: Use only position 1 (bitmap 0100...) . ALL: Use all possible positions (bitmap 1111...) . UDEFined: See [CONFigure:]SIGNaling:NRADio:CELL:SSB:BEAM:MODel.

6.3.4.10.2.586 TciStates

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:SSB:BEAM:TCIStates:UPDate
```

class TciStatesCls

TciStates commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set_update(*cell_name: str*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SSB:BEAM:TCIStates:UPDate
driver.configure.signaling.nradio.cell.ssb.beam.tciStates.set_update(cell_name␣
↪= 'abc')
```

Updates transmission configuration indicator (TCI) states.

param cell_name
No help available

6.3.4.10.2.587 HfOffset

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:SSB:HFOFfset
```

class HfOffsetCls

HfOffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SSB:HFOFfset
value: int = driver.configure.signaling.nradio.cell.ssb.hfOffset.get(cell_name␣
↪= 'abc')
```

Adds an offset to the SSB transmission timing (number of half frames) .

param cell_name
No help available

return
offset: No help available

set(*cell_name: str*, *offset: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SSB:HFOFfset
driver.configure.signaling.nradio.cell.ssb.hfOffset.set(cell_name = 'abc',␣
↪offset = 1)
```

Adds an offset to the SSB transmission timing (number of half frames) .

param cell_name
No help available

param offset
No help available

6.3.4.10.2.588 PaOffset

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:SSB:PAOFfset
```

class PaOffsetCls

PaOffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SSB:PAOFfset
value: int = driver.configure.signaling.nradio.cell.ssb.paOffset.get(cell_name␣
↪= 'abc')
```

Defines the parameter 'offsetToPointA' of the SIB.

param cell_name
No help available

return
offset: Number of RB

set(*cell_name: str*, *offset: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SSB:PAOFfset
driver.configure.signaling.nradio.cell.ssb.paOffset.set(cell_name = 'abc',␣
↪offset = 1)
```

Defines the parameter 'offsetToPointA' of the SIB.

param cell_name
No help available

param offset
Number of RB

6.3.4.10.2.589 Periodicity

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:SSB:PERiodicity
```

class PeriodicityCls

Periodicity commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → PeriodicityB

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SSB:PERiodicity
value: enums.PeriodicityB = driver.configure.signaling.nradio.cell.ssb.
↪periodicity.get(cell_name = 'abc')
```

Selects the periodicity of the SSB in the time domain.

param cell_name
No help available

return
periodicity: Periodicity in ms (5 ms to 160 ms)

set(*cell_name: str*, *periodicity: PeriodicityB*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SSB:PERiodicity
driver.configure.signaling.nradio.cell.ssb.periodicity.set(cell_name = 'abc',␣
↪periodicity = enums.PeriodicityB.P10)
```

Selects the periodicity of the SSB in the time domain.

param cell_name
No help available

param periodicity
Periodicity in ms (5 ms to 160 ms)

6.3.4.10.2.590 Soffset

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:SSB:SOFFset
```

class SoffsetCls

Soffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SSB:SOFFset
value: int = driver.configure.signaling.nradio.cell.ssb.soffset.get(cell_name =
↪'abc')
```

Defines the kSSB, the number of SC between the SSB and the overall RB grid.

param cell_name
No help available

return
offset: Number of SC

set(*cell_name: str*, *offset: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SSB:SOFFset
driver.configure.signaling.nradio.cell.ssb.soffset.set(cell_name = 'abc',␣
↪offset = 1)
```

Defines the kSSB, the number of SC between the SSB and the overall RB grid.

param cell_name
No help available

param offset
Number of SC

6.3.4.10.2.591 Sspacing

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:SSB:SSPacing
```

class SspacingCls

Sspacing commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → SubCarrSpacing

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SSB:SSPacing
value: enums.SubCarrSpacing = driver.configure.signaling.nradio.cell.ssb.
↪sspacing.get(cell_name = 'abc')
```

Selects the SSB case and thus the SSB subcarrier spacing.

param cell_name
No help available

return
spacing: Case A to case E plus related subcarrier spacing in kHz Example: A15 = case A, 15 kHz

set(*cell_name: str*, *spacing: SubCarrSpacing*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SSB:SSPacing
driver.configure.signaling.nradio.cell.ssb.sspacing.set(cell_name = 'abc',␣
↪spacing = enums.SubCarrSpacing.A15)
```

Selects the SSB case and thus the SSB subcarrier spacing.

param cell_name
No help available

param spacing
Case A to case E plus related subcarrier spacing in kHz Example: A15 = case A, 15 kHz

6.3.4.10.2.592 Transmission

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:SSB:TRANsmission
```

class TransmissionCls

Transmission commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → AutoMode

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SSB:TRANsmission
value: enums.AutoMode = driver.configure.signaling.nradio.cell.ssb.transmission.
↪get(cell_name = 'abc')
```

No command help available

param cell_name
No help available

return
burst_trx: No help available

set(*cell_name: str*, *burst_trx: AutoMode*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SSB:TRANsmission
driver.configure.signaling.nradio.cell.ssb.transmission.set(cell_name = 'abc',␣
↪burst_trx = enums.AutoMode.AUTO)
```

No command help available

param cell_name
No help available

param burst_trx
No help available

6.3.4.10.2.593 Sspacing

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:SSPacing
```

class SspacingCls

Sspacing commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SSPacing
value: int = driver.configure.signaling.nradio.cell.sspacing.get(cell_name =
↪'abc')
```

Configures the subcarrier spacing for the initial BWP. The setting is coupled to the common SCS setting, see [CONFigure:]SIGNaling:NRADio:CELL:RFSettings:SSPacing.

param cell_name
No help available

return
spacing: No help available

set(*cell_name: str*, *spacing: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:SSPacing
driver.configure.signaling.nradio.cell.sspacing.set(cell_name = 'abc', spacing
↪= 1)
```

Configures the subcarrier spacing for the initial BWP. The setting is coupled to the common SCS setting, see [CONFigure:]SIGNaling:NRADio:CELL:RFSettings:SSPacing.

param cell_name
No help available

param spacing
No help available

6.3.4.10.2.594 Tadvance

class TadvanceCls

Tadvance commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.tadvance.clone()
```

Subgroups

6.3.4.10.2.595 Periodicity

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:TADVance:PERiodicity
```

class PeriodicityCls

Periodicity commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → TadvPeriodicity

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:TADVance:PERiodicity
value: enums.TadvPeriodicity = driver.configure.signaling.nradio.cell.tadvance.
↪periodicity.get(cell_name = 'abc')
```

Configures the periodicity for sending timing advance commands to the UE, for the initial BWP.

param cell_name
No help available

return
periodicity: No help available

set(*cell_name: str*, *periodicity: TadvPeriodicity*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:TADVance:PERiodicity
driver.configure.signaling.nradio.cell.tadvance.periodicity.set(cell_name = 'abc
↪', periodicity = enums.TadvPeriodicity.CONTinuous)
```

Configures the periodicity for sending timing advance commands to the UE, for the initial BWP.

param cell_name
No help available

param periodicity
No help available

6.3.4.10.2.596 Timing

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:TADVance:TIMing
```

class TimingCls

Timing commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:TADVance:TIMing
value: int = driver.configure.signaling.nradio.cell.tadvance.timing.get(cell_
↪name = 'abc')
```

Configures a timing advance value to be sent to the UE, for the initial BWP.

param cell_name
No help available

return
ta: No help available

set(*cell_name: str*, *ta: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:TADVance:TIMing
driver.configure.signaling.nradio.cell.tadvance.timing.set(cell_name = 'abc',␣
↪ta = 1)
```

Configures a timing advance value to be sent to the UE, for the initial BWP.

param cell_name
No help available

param ta
No help available

6.3.4.10.2.597 Tdd

class TddCls

Tdd commands group definition. 8 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.tdd.clone()
```

Subgroups

6.3.4.10.2.598 Enable

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:TDD:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:TDD:ENABle
value: bool = driver.configure.signaling.nradio.cell.tdd.enable.get(cell_name =
↪'abc')
```

Enables or disables signaling of the 'TDD-UL-DL-ConfigCommon' to the UE.

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:TDD:ENABle
driver.configure.signaling.nradio.cell.tdd.enable.set(cell_name = 'abc', enable␣
↪= False)
```

Enables or disables signaling of the 'TDD-UL-DL-ConfigCommon' to the UE.

param cell_name
No help available

param enable
No help available

6.3.4.10.2.599 Pattern<Pattern>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.configure.signaling.nradio.cell.tdd.pattern.repcap_pattern_get()
driver.configure.signaling.nradio.cell.tdd.pattern.repcap_pattern_set(repcap.Pattern.Nr1)
```

class PatternCls

Pattern commands group definition. 6 total commands, 4 Subgroups, 0 group commands Repeated Capability: Pattern, default value after init: Pattern.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.tdd.pattern.clone()
```

Subgroups

6.3.4.10.2.600 Downlink

class DownlinkCls

Downlink commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.tdd.pattern.downlink.clone()
```

Subgroups

6.3.4.10.2.601 FsSymbol

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:TDD:PATTern<PatternNo>:DL:FSSYmbol
```

class FsSymbolCls

FsSymbol commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *pattern=Pattern.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:TDD:PATTern<PatternNo>:DL:FSSYmbol
value: int = driver.configure.signaling.nradio.cell.tdd.pattern.downlink.
↪fsSymbol.get(cell_name = 'abc', pattern = repcap.Pattern.Default)
```

Defines the number of DL symbols in the partial slot of the UL-DL pattern {p}.

param cell_name
No help available

param pattern
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pattern')

return
symbol: No help available

set(*cell_name: str*, *symbol: int*, *pattern=Pattern.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:TDD:PATTern<PatternNo>:DL:FSSYmbol
driver.configure.signaling.nradio.cell.tdd.pattern.downlink.fsSymbol.set(cell_
↪name = 'abc', symbol = 1, pattern = repcap.Pattern.Default)
```

Defines the number of DL symbols in the partial slot of the UL-DL pattern {p}.

param cell_name
No help available

param symbol
No help available

param pattern
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pattern')

6.3.4.10.2.602 Nslots

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:TDD:PATTern<PatternNo>:DL:NSLots
```

class NslotsCls

Nslots commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *pattern=Pattern.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:TDD:PATTern<PatternNo>:DL:NSLots
value: int = driver.configure.signaling.nradio.cell.tdd.pattern.downlink.nslots.
↪get(cell_name = 'abc', pattern = repcap.Pattern.Default)
```

Defines the number of full DL slots in the UL-DL pattern {p}.

param cell_name
No help available

param pattern
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pattern')

return
number: No help available

set(*cell_name: str*, *number: int*, *pattern=Pattern.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:TDD:PATTern<PatternNo>:DL:NSLots
driver.configure.signaling.nradio.cell.tdd.pattern.downlink.nslots.set(cell_
↪name = 'abc', number = 1, pattern = repcap.Pattern.Default)
```

Defines the number of full DL slots in the UL-DL pattern {p}.

param cell_name
No help available

param number
No help available

param pattern
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pattern')

6.3.4.10.2.603 Enable

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:TDD:PATTern<PatternNo>:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *pattern=Pattern.Default*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:TDD:PATTern<PatternNo>:ENABle
value: bool = driver.configure.signaling.nradio.cell.tdd.pattern.enable.
↪get(cell_name = 'abc', pattern = repcap.Pattern.Default)
```

Enables or disables the second UL-DL pattern.

param cell_name
No help available

param pattern
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pattern')

return
enable: No help available

set(*cell_name: str*, *enable: bool*, *pattern=Pattern.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:TDD:PATTern<PatternNo>:ENABle
driver.configure.signaling.nradio.cell.tdd.pattern.enable.set(cell_name = 'abc',
↪ enable = False, pattern = repcap.Pattern.Default)
```

Enables or disables the second UL-DL pattern.

param cell_name
No help available

param enable
No help available

param pattern
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pattern')

6.3.4.10.2.604 Periodicity

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:TDD:PATTern<PatternNo>:PERiodicity
```

class PeriodicityCls

Periodicity commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *pattern=Pattern.Default*) → Periodicity

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:TDD:PATTern<PatternNo>:PERiodicity
value: enums.Periodicity = driver.configure.signaling.nradio.cell.tdd.pattern.
↪periodicity.get(cell_name = 'abc', pattern = repcap.Pattern.Default)
```

Configures the periodicity of the UL-DL pattern {p}.

param cell_name
No help available

param pattern
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pattern')

return
periodicity: Periodicity, 0.5 ms (P0P5) to 10 ms (P10) .

set(*cell_name: str*, *periodicity: Periodicity*, *pattern=Pattern.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:TDD:PATTern<PatternNo>:PERiodicity
driver.configure.signaling.nradio.cell.tdd.pattern.periodicity.set(cell_name =
↪'abc', periodicity = enums.Periodicity.P0P5, pattern = repcap.Pattern.Default)
```

Configures the periodicity of the UL-DL pattern {p}.

param cell_name
No help available

param periodicity
Periodicity, 0.5 ms (P0P5) to 10 ms (P10) .

param pattern
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pattern')

6.3.4.10.2.605 Uplink

class UplinkCls

Uplink commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.tdd.pattern.uplink.clone()
```

Subgroups

6.3.4.10.2.606 FsSymbol

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:TDD:PATTern<PatternNo>:UL:FSSYmbol
```

class FsSymbolCls

FsSymbol commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *pattern=Pattern.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:TDD:PATTern<PatternNo>:UL:FSSYmbol
value: int = driver.configure.signaling.nradio.cell.tdd.pattern.uplink.fsSymbol.
↪get(cell_name = 'abc', pattern = repcap.Pattern.Default)
```

Defines the number of UL symbols in the partial slot of the UL-DL pattern {p}.

param cell_name
No help available

param pattern
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pattern')

return
symbol: No help available

set(*cell_name: str*, *symbol: int*, *pattern=Pattern.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:TDD:PATTern<PatternNo>:UL:FSSYmbol
driver.configure.signaling.nradio.cell.tdd.pattern.uplink.fsSymbol.set(cell_
↪name = 'abc', symbol = 1, pattern = repcap.Pattern.Default)
```

Defines the number of UL symbols in the partial slot of the UL-DL pattern {p}.

param cell_name
No help available

param symbol
No help available

param pattern
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pattern')

6.3.4.10.2.607 Nslots

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:TDD:PATTern<PatternNo>:UL:NSLots
```

class NslotsCls

Nslots commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *pattern=Pattern.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:TDD:PATTern<PatternNo>:UL:NSLots
value: int = driver.configure.signaling.nradio.cell.tdd.pattern.uplink.nslots.
↪get(cell_name = 'abc', pattern = repcap.Pattern.Default)
```

Defines the number of full UL slots in the UL-DL pattern {p}.

param cell_name
No help available

param pattern
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pattern')

return
number: No help available

set(*cell_name: str*, *number: int*, *pattern=Pattern.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:TDD:PATTern<PatternNo>:UL:NSLots
driver.configure.signaling.nradio.cell.tdd.pattern.uplink.nslots.set(cell_name␣
↪= 'abc', number = 1, pattern = repcap.Pattern.Default)
```

Defines the number of full UL slots in the UL-DL pattern {p}.

param cell_name
No help available

param number
No help available

param pattern
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pattern')

6.3.4.10.2.608 Uplink

class UplinkCls

Uplink commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.tdd.uplink.clone()
```

Subgroups

6.3.4.10.2.609 MdCycle

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:TDD:UL:MDCYcle
```

class MdCycleCls

MdCycle commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → UlMaxDutyCyle

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:TDD:UL:MDCYcle
value: enums.UlMaxDutyCyle = driver.configure.signaling.nradio.cell.tdd.uplink.
↪mdCycle.get(cell_name = 'abc')
```

Configures the maximum percentage of scheduled UL symbols. Selecting a D8x value also configures other settings so that they are compatible to maximum UL duty cycle tests.

param cell_name
No help available

return
ul_max_duty_cyle: OFF: no maximum duty cycle applied D80 | D82 | D85 | D87: 80 % to 87 % D89: 89.6 %

set(*cell_name: str*, *ul_max_duty_cyle: UlMaxDutyCyle*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:TDD:UL:MDCYcle
driver.configure.signaling.nradio.cell.tdd.uplink.mdCycle.set(cell_name = 'abc',
↪ ul_max_duty_cyle = enums.UlMaxDutyCyle.D80)
```

Configures the maximum percentage of scheduled UL symbols. Selecting a D8x value also configures other settings so that they are compatible to maximum UL duty cycle tests.

param cell_name
No help available

param ul_max_duty_cyle
OFF: no maximum duty cycle applied D80 | D82 | D85 | D87: 80 % to 87 % D89: 89.6 %

6.3.4.10.2.610 Timeout

class TimeoutCls

Timeout commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.timeout.clone()
```

Subgroups

6.3.4.10.2.611 N<Nnum>

RepCap Settings

```
# Range: Nr310 .. Nr311
rc = driver.configure.signaling.nradio.cell.timeout.n.repcap_nnum_get()
driver.configure.signaling.nradio.cell.timeout.n.repcap_nnum_set(repcap.Nnum.Nr310)
```

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:TOUT:N<no>
```

class NCls

N commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Nnum, default value after init: Nnum.Nr310

get(*cell_name: str*, *nnum=Nnum.Default*) → Counter

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:TOUT:N<no>
value: enums.Counter = driver.configure.signaling.nradio.cell.timeout.n.
↪get(cell_name = 'abc', nnum = repcap.Nnum.Default)


    INTRO_CMD_HELP: Configures one of the following constants:

    - N310, for counting out-of-sync indications and starting T310
    - N311, for counting in-sync indications and stopping T310

    :param cell_name: No help available
    :param nnum: optional repeated capability selector. Default value: Nr310␣
↪(settable in the interface 'N')
    :return: counter: Value of the constant For N310, the value N5 is not␣
↪allowed. For N311, the value N20 is not allowed.
```

set(*cell_name: str*, *counter: Counter*, *nnum=Nnum.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:TOUT:N<no>
driver.configure.signaling.nradio.cell.timeout.n.set(cell_name = 'abc', counter
↪= enums.Counter.N1, nnum = repcap.Nnum.Default)


    INTRO_CMD_HELP: Configures one of the following constants:

    - N310, for counting out-of-sync indications and starting T310
    - N311, for counting in-sync indications and stopping T310

    :param cell_name: No help available
    :param counter: Value of the constant For N310, the value N5 is not allowed.
↪ For N311, the value N20 is not allowed.
    :param nnum: optional repeated capability selector. Default value: Nr310
↪(settable in the interface 'N')
```

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.timeout.n.clone()
```

6.3.4.10.2.612 T<Tnum>

RepCap Settings

```
# Range: Nr300 .. Nr319
rc = driver.configure.signaling.nradio.cell.timeout.t.repcap_tnum_get()
driver.configure.signaling.nradio.cell.timeout.t.repcap_tnum_set(repcap.Tnum.Nr300)
```

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:TOUT:T<no>
```

class TCls

T commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Tnum, default value after init: Tnum.Nr300

get(*cell_name: str*, *tnum=Tnum.Default*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:TOUT:T<no>
value: int = driver.configure.signaling.nradio.cell.timeout.t.get(cell_name =
↪'abc', tnum = repcap.Tnum.Default)


    INTRO_CMD_HELP: Configures one of the following timers:

    - T300, RRC connection establishment
    - T301, RRC connection re-establishment, after cell selection
```

(continues on next page)

(continued from previous page)

```
    - T310, detection of radio link failure (out-of-sync)
    - T311, RRC connection re-establishment, before cell selection
    - T319, RRC connection resume

    :param cell_name: No help available
    :param tnum: optional repeated capability selector. Default value: Nr300␣
↪(settable in the interface 'T')
    :return: timer: Timeout value
```

set(*cell_name: str*, *timer: int*, *tnum=Tnum.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:TOUT:T<no>
driver.configure.signaling.nradio.cell.timeout.t.set(cell_name = 'abc', timer =␣
↪1, tnum = repcap.Tnum.Default)


    INTRO_CMD_HELP: Configures one of the following timers:

    - T300, RRC connection establishment
    - T301, RRC connection re-establishment, after cell selection
    - T310, detection of radio link failure (out-of-sync)
    - T311, RRC connection re-establishment, before cell selection
    - T319, RRC connection resume

    :param cell_name: No help available
    :param timer: Timeout value
    :param tnum: optional repeated capability selector. Default value: Nr300␣
↪(settable in the interface 'T')
```

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.timeout.t.clone()
```

6.3.4.10.2.613 Timing

class TimingCls

Timing commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.timing.clone()
```

Subgroups

6.3.4.10.2.614 DltShift

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:TIMing:DLTShift
```

class DltShiftCls

DltShift commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:TIMing:DLTShift
value: int = driver.configure.signaling.nradio.cell.timing.dltShift.get(cell_
↪name = 'abc')
```

Defines a DL time offset for a cell, after switching it on.

param cell_name
No help available

return
delta: No help available

set(*cell_name: str*, *delta: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:TIMing:DLTShift
driver.configure.signaling.nradio.cell.timing.dltShift.set(cell_name = 'abc',␣
↪delta = 1)
```

Defines a DL time offset for a cell, after switching it on.

param cell_name
No help available

param delta
No help available

6.3.4.10.2.615 Offset

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:TIMing:OFFSet
```

class OffsetCls

Offset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:TIMing:OFFSet
value: int = driver.configure.signaling.nradio.cell.timing.offset.get(cell_name␣
↪= 'abc')
```

Defines a time offset for a cell, before switching it on.

param cell_name
No help available

return
time_offset: No help available

set(*cell_name: str*, *time_offset: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:TIMing:OFFSet
driver.configure.signaling.nradio.cell.timing.offset.set(cell_name = 'abc',␣
↪time_offset = 1)
```

Defines a time offset for a cell, before switching it on.

param cell_name
No help available

param time_offset
No help available

6.3.4.10.2.616 SfnOffset

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:TIMing:SFNoffset
```

class SfnOffsetCls

SfnOffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:TIMing:SFNoffset
value: int = driver.configure.signaling.nradio.cell.timing.sfnOffset.get(cell_
↪name = 'abc')
```

Defines a system frame number offset for a cell, before switching it on.

param cell_name
No help available

return
sfn_offset: No help available

set(*cell_name: str*, *sfn_offset: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:TIMing:SFNoffset
driver.configure.signaling.nradio.cell.timing.sfnOffset.set(cell_name = 'abc',␣
↪sfn_offset = 1)
```

Defines a system frame number offset for a cell, before switching it on.

param cell_name
No help available

param sfn_offset
No help available

6.3.4.10.2.617 UeScheduling

class UeSchedulingCls

UeScheduling commands group definition. 77 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.ueScheduling.clone()
```

Subgroups

6.3.4.10.2.618 CmMapping

class CmMappingCls

CmMapping commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.ueScheduling.cmMapping.clone()
```

Subgroups

6.3.4.10.2.619 Mcs

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:CMMapping:MCS
```

class McsCls

Mcs commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → List[int]

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:CMMapping:MCS
value: List[int] = driver.configure.signaling.nradio.cell.ueScheduling.
↪cmMapping.mcs.get(cell_name = 'abc')
```

Sets the configuration mode to UDEFined and defines the mapping table for that mode, for the initial BWP. A query returns the mapping table contents for the currently used configuration mode, without changing the mode. For setting the configuration mode, see [CONFigure:]SIGNaling:NRADio:CELL:UESCheduling:CMMapping:MCSTable.

param cell_name
No help available

return
mcs: Comma-separated list of 16 MCS values, for reported CQI values 0 to 15.

set(*cell_name: str*, *mcs: List[int]*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UEScheduling:CMMapping:MCS
driver.configure.signaling.nradio.cell.ueScheduling.cmMapping.mcs.set(cell_name␣
↪= 'abc', mcs = [1, 2, 3])
```

Sets the configuration mode to UDEFined and defines the mapping table for that mode, for the initial BWP. A query returns the mapping table contents for the currently used configuration mode, without changing the mode. For setting the configuration mode, see [CONFigure:]SIGNaling:NRADio:CELL:UEScheduling:CMMapping:MCSTable.

param cell_name
No help available

param mcs
Comma-separated list of 16 MCS values, for reported CQI values 0 to 15.

6.3.4.10.2.620 McsTable

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UEScheduling:CMMapping:MCSTable
```

class McsTableCls

McsTable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- **Mcs_Table: enums.McsTableC:**
  - AUTO: The mapping table is selected automatically, depending on [CONFigure:]SIGNaling:NRADio:CELL:UEScheduling:UDEFined:DL:MCSTable.
  - P521: The mapping table contents are defined by 3GPP TS 38.521-4.Table selection via Predefined3GPP.
  - UDEFined: The mapping table contents are defined via a separate command, see [CONFigure:]SIGNaling:NRADio:CELL:UEScheduling:CMMapping:MCS.
- Predefined_3_Gpp: enums.ConfigTypeB: Selects a mapping table for MCSTable = P521. T1: table A.4-1 in 3GPP TS 38.

521-4 T2: table A.4-2 in 3GPP TS 38.521-4 T3: table A.4-3 in 3GPP TS 38.521-4

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UEScheduling:CMMapping:MCSTable
value: GetStruct = driver.configure.signaling.nradio.cell.ueScheduling.
↪cmMapping.mcsTable.get(cell_name = 'abc')
```

Selects a configuration mode for the CQI-MCS mapping table for follow WB CQI, for the initial BWP.

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *mcs_table: McsTableC*, *predefined_3_gpp: ConfigTypeB = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:CMMapping:MCSTable
driver.configure.signaling.nradio.cell.ueScheduling.cmMapping.mcsTable.set(cell_
↪name = 'abc', mcs_table = enums.McsTableC.AUTO, predefined_3_gpp = enums.
↪ConfigTypeB.T1)
```

Selects a configuration mode for the CQI-MCS mapping table for follow WB CQI, for the initial BWP.

param cell_name
No help available

param mcs_table

- AUTO: The mapping table is selected automatically, depending on [CONFigure:]SIGNaling:NRADio:CELL:UESCheduling:UDEFined:DL:MCSTable.
- P521: The mapping table contents are defined by 3GPP TS 38.521-4.Table selection via Predefined3GPP.
- UDEFined: The mapping table contents are defined via a separate command, see [CONFigure:]SIGNaling:NRADio:CELL:UESCheduling:CMMapping:MCS.

param predefined_3_gpp
Selects a mapping table for MCSTable = P521. T1: table A.4-1 in 3GPP TS 38.521-4 T2: table A.

4-2 in 3GPP TS 38.521-4 T3: table A.4-3 in 3GPP TS 38.521-4

6.3.4.10.2.621 Rmc

class RmcCls

Rmc commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.ueScheduling.rmc.clone()
```

Subgroups

6.3.4.10.2.622 Downlink

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:RMC:DL
```

class DownlinkCls

Downlink commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Enable: bool: Enables or disables scheduling for all DL slots.

- Modulation: enums.ModulationB: QPSK, 16QAM, 64QAM, 256QAM
- Number_Rb: int: No parameter help available
- Start_Rb: int: No parameter help available

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UEScheduling:RMC:DL
value: GetStruct = driver.configure.signaling.nradio.cell.ueScheduling.rmc.
↪downlink.get(cell_name = 'abc')
```

Configures NR cell settings to values compliant with a DL RMC definition. A setting command accepts only certain value combinations. Use the RMC wizard in the GUI to get allowed value combinations. A query returns the set of values that is presented by the RMC wizard. These values can differ from currently applied values. Omit optional parameters only if you do not care which value you get (just any RMC-compliant value) .

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *enable: bool*, *modulation: ModulationB = None*, *number_rb: int = None*, *start_rb: int = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UEScheduling:RMC:DL
driver.configure.signaling.nradio.cell.ueScheduling.rmc.downlink.set(cell_name␣
↪= 'abc', enable = False, modulation = enums.ModulationB.BPSK, number_rb = 1,␣
↪start_rb = 1)
```

Configures NR cell settings to values compliant with a DL RMC definition. A setting command accepts only certain value combinations. Use the RMC wizard in the GUI to get allowed value combinations. A query returns the set of values that is presented by the RMC wizard. These values can differ from currently applied values. Omit optional parameters only if you do not care which value you get (just any RMC-compliant value) .

param cell_name
No help available

param enable
Enables or disables scheduling for all DL slots.

param modulation
QPSK, 16QAM, 64QAM, 256QAM

param number_rb
No help available

param start_rb
No help available

6.3.4.10.2.623 Uplink

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UEScheduling:RMC:UL
```

class UplinkCls

Uplink commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Enable: bool: Enables or disables scheduling for all UL slots.
- Modulation: enums.ModulationB: /2-BPSK, QPSK, 16QAM, 64QAM, 256QAM
- Number_Rb: int: No parameter help available
- Start_Rb: int: No parameter help available
- Tp_Recoding: enums.Waveform: OFDM type CP-OFDM or DFT-s-OFDM

class SetStruct

Structure for setting input parameters. Contains optional setting parameters. Fields:

- Cell_Name: str: No parameter help available
- Enable: bool: Enables or disables scheduling for all UL slots.
- Modulation: enums.ModulationB: Optional setting parameter. /2-BPSK, QPSK, 16QAM, 64QAM, 256QAM
- Number_Rb: int: No parameter help available
- Start_Rb: int: No parameter help available
- Tp_Recoding: enums.Waveform: Optional setting parameter. OFDM type CP-OFDM or DFT-s-OFDM

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UEScheduling:RMC:UL
value: GetStruct = driver.configure.signaling.nradio.cell.ueScheduling.rmc.
↪uplink.get(cell_name = 'abc')
```

Configures NR cell settings to values compliant with a UL RMC definition. A setting command accepts only certain value combinations. Use the RMC wizard in the GUI to get allowed value combinations. A query returns the set of values that is presented by the RMC wizard. These values can differ from currently applied values. Omit optional parameters only if you do not care which value you get (just any RMC-compliant value) .

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*structure: SetStruct*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:RMC:UL
structure = driver.configure.signaling.nradio.cell.ueScheduling.rmc.uplink.
↪SetStruct()
structure.Cell_Name: str = 'abc'
structure.Enable: bool = False
structure.Modulation: enums.ModulationB = enums.ModulationB.BPSK
structure.Number_Rb: int = 1
structure.Start_Rb: int = 1
structure.Tp_Recoding: enums.Waveform = enums.Waveform.CP
driver.configure.signaling.nradio.cell.ueScheduling.rmc.uplink.set(structure)
```

Configures NR cell settings to values compliant with a UL RMC definition. A setting command accepts only certain value combinations. Use the RMC wizard in the GUI to get allowed value combinations. A query returns the set of values that is presented by the RMC wizard. These values can differ from currently applied values. Omit optional parameters only if you do not care which value you get (just any RMC-compliant value) .

param structure
for set value, see the help for SetStruct structure arguments.

6.3.4.10.2.624 Smode

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SMODe
```

class SmodeCls

Smode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → ModeUeScheduling

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SMODe
value: enums.ModeUeScheduling = driver.configure.signaling.nradio.cell.
↪ueScheduling.smode.get(cell_name = 'abc')
```

Selects a scheduling mode for DL and UL, for the initial BWP.

param cell_name
No help available

return
mode: FIXed: Fixed scheduling, DL and UL. SPS: SPS DL, CG UL. CQI: Follow CQI WB DL, fixed scheduling UL. PRI: Follow PMI WB + RI DL, fixed scheduling UL. CPRI: Follow CQI WB + PMI WB + RI DL, fixed scheduling UL. BO: Follow buffer occupancy (BO) DL, fixed scheduling UL. UDEFined: Other dynamic scheduling mode (query only) .

set(*cell_name: str*, *mode: ModeUeScheduling*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SMODe
driver.configure.signaling.nradio.cell.ueScheduling.smode.set(cell_name = 'abc',
↪ mode = enums.ModeUeScheduling.BO)
```

Selects a scheduling mode for DL and UL, for the initial BWP.

param cell_name
No help available

param mode
FIXed: Fixed scheduling, DL and UL. SPS: SPS DL, CG UL. CQI: Follow CQI WB DL, fixed scheduling UL. PRI: Follow PMI WB + RI DL, fixed scheduling UL. CPRI: Follow CQI WB + PMI WB + RI DL, fixed scheduling UL. BO: Follow buffer occupancy (BO) DL, fixed scheduling UL. UDEFined: Other dynamic scheduling mode (query only) .

6.3.4.10.2.625 Sps

class SpsCls

Sps commands group definition. 36 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.ueScheduling.sps.clone()
```

Subgroups

6.3.4.10.2.626 Downlink

class DownlinkCls

Downlink commands group definition. 10 total commands, 10 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.ueScheduling.sps.downlink.clone()
```

Subgroups

6.3.4.10.2.627 Alevel

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:DL:ALEVel
```

class AlevelCls

Alevel commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → Level

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:DL:ALEVel
value: enums.Level = driver.configure.signaling.nradio.cell.ueScheduling.sps.
↪downlink.alevel.get(cell_name = 'abc')
```

Configures the aggregation level for DL SPS scheduling, for the initial BWP.

param cell_name
No help available

return
alevel: No help available

set(*cell_name: str*, *alevel: Level*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:DL:ALEVel
driver.configure.signaling.nradio.cell.ueScheduling.sps.downlink.alevel.
↪set(cell_name = 'abc', alevel = enums.Level.AL1)
```

Configures the aggregation level for DL SPS scheduling, for the initial BWP.

param cell_name
No help available

param alevel
No help available

6.3.4.10.2.628 All

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:DL:ALL
```

class AllCls

All commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Periodicity: int: No parameter help available
- Mcs_Table: enums.McsTableB: 256QAM, 64QAM low SE, 64QAM
- Alevel: enums.Level: Aggregation level
- Search_Space_Id: int: No parameter help available
- Resource_Allocation_Type: enums.ResourceAllocationType: No parameter help available
- Rgb_Size: enums.RgbSize: Config 1 or 2
- No_Harq: int: Signaled 'nrofHARQ-Processes'
- Mapping: enums.MappingI: Interleaved or non-interleaved virtual RB to physical RB mapping
- Padding: enums.SpsPadding: No DL padding or with DL padding

class SetStruct

Structure for setting input parameters. Contains optional setting parameters. Fields:

- Cell_Name: str: No parameter help available
- Periodicity: int: No parameter help available
- Mcs_Table: enums.McsTableB: Optional setting parameter. 256QAM, 64QAM low SE, 64QAM
- Alevel: enums.Level: Optional setting parameter. Aggregation level

- Search_Space_Id: int: No parameter help available
- Resource_Allocation_Type: enums.ResourceAllocationType: No parameter help available
- Rgb_Size: enums.RgbSize: Optional setting parameter. Config 1 or 2
- No_Harq: int: Optional setting parameter. Signaled 'nrofHARQ-Processes'
- Mapping: enums.MappingI: Optional setting parameter. Interleaved or non-interleaved virtual RB to physical RB mapping
- Padding: enums.SpsPadding: Optional setting parameter. No DL padding or with DL padding

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UEScheduling:SPS:DL:ALL
value: GetStruct = driver.configure.signaling.nradio.cell.ueScheduling.sps.
↪downlink.all.get(cell_name = 'abc')
```

Configures several settings for SPS DL scheduling (combination of the other ...SPS:DL:... commands) , for the initial BWP.

param cell_name
Type 0, type 1, dynamic switch

return
structure: for return value, see the help for GetStruct structure arguments.

set(*structure: SetStruct*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UEScheduling:SPS:DL:ALL
structure = driver.configure.signaling.nradio.cell.ueScheduling.sps.downlink.
↪all.SetStruct()
structure.Cell_Name: str = 'abc'
structure.Periodicity: int = 1
structure.Mcs_Table: enums.McsTableB = enums.McsTableB.L64
structure.Alevel: enums.Level = enums.Level.AL1
structure.Search_Space_Id: int = 1
structure.Resource_Allocation_Type: enums.ResourceAllocationType = enums.
↪ResourceAllocationType.DSWich
structure.Rgb_Size: enums.RgbSize = enums.RgbSize.CON1
structure.No_Harq: int = 1
structure.Mapping: enums.MappingI = enums.MappingI.INT
structure.Padding: enums.SpsPadding = enums.SpsPadding.ALLZero
driver.configure.signaling.nradio.cell.ueScheduling.sps.downlink.all.
↪set(structure)
```

Configures several settings for SPS DL scheduling (combination of the other ...SPS:DL:... commands) , for the initial BWP.

param structure
for set value, see the help for SetStruct structure arguments.

6.3.4.10.2.629 Mapping

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:DL:MAPPing
```

class MappingCls

Mapping commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → MappingI

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:DL:MAPPing
value: enums.MappingI = driver.configure.signaling.nradio.cell.ueScheduling.sps.
↪downlink.mapping.get(cell_name = 'abc')
```

Selects whether interleaved or non-interleaved virtual RB to physical RB mapping is applied for the PDSCH, for DL SPS scheduling, for the initial BWP.

param cell_name
No help available

return
mapping: Interleaved or non-interleaved

set(*cell_name: str*, *mapping: MappingI*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:DL:MAPPing
driver.configure.signaling.nradio.cell.ueScheduling.sps.downlink.mapping.
↪set(cell_name = 'abc', mapping = enums.MappingI.INT)
```

Selects whether interleaved or non-interleaved virtual RB to physical RB mapping is applied for the PDSCH, for DL SPS scheduling, for the initial BWP.

param cell_name
No help available

param mapping
Interleaved or non-interleaved

6.3.4.10.2.630 McsTable

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:DL:MCSTable
```

class McsTableCls

McsTable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → McsTableB

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:DL:MCSTable
value: enums.McsTableB = driver.configure.signaling.nradio.cell.ueScheduling.
↪sps.downlink.mcsTable.get(cell_name = 'abc')
```

Configures the signaled mcs-Table for DL SPS scheduling, for the initial BWP.

param cell_name
No help available

return
mcs_table: 256QAM, 64QAM low SE, 64QAM

set(*cell_name: str*, *mcs_table: McsTableB*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UEScheduling:SPS:DL:MCSTable
driver.configure.signaling.nradio.cell.ueScheduling.sps.downlink.mcsTable.
↪set(cell_name = 'abc', mcs_table = enums.McsTableB.L64)
```

Configures the signaled mcs-Table for DL SPS scheduling, for the initial BWP.

param cell_name
No help available

param mcs_table
256QAM, 64QAM low SE, 64QAM

6.3.4.10.2.631 Nohp

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UEScheduling:SPS:DL:NOHP
```

class NohpCls

Nohp commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UEScheduling:SPS:DL:NOHP
value: int = driver.configure.signaling.nradio.cell.ueScheduling.sps.downlink.
↪nohp.get(cell_name = 'abc')
```

Configures the signaled 'nrofHARQ-Processes' for DL SPS scheduling, for the initial BWP.

param cell_name
No help available

return
no_harq: No help available

set(*cell_name: str*, *no_harq: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UEScheduling:SPS:DL:NOHP
driver.configure.signaling.nradio.cell.ueScheduling.sps.downlink.nohp.set(cell_
↪name = 'abc', no_harq = 1)
```

Configures the signaled 'nrofHARQ-Processes' for DL SPS scheduling, for the initial BWP.

param cell_name
No help available

param no_harq
No help available

6.3.4.10.2.632 Padding

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:DL:PADDing
```

class PaddingCls

Padding commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → SpsPadding

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:DL:PADDing
value: enums.SpsPadding = driver.configure.signaling.nradio.cell.ueScheduling.
↪sps.downlink.padding.get(cell_name = 'abc')
```

Activates or deactivates downlink padding for SPS scheduling, for the initial BWP.

param cell_name
No help available

return
padding: No help available

set(*cell_name: str*, *padding: SpsPadding*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:DL:PADDing
driver.configure.signaling.nradio.cell.ueScheduling.sps.downlink.padding.
↪set(cell_name = 'abc', padding = enums.SpsPadding.ALLZero)
```

Activates or deactivates downlink padding for SPS scheduling, for the initial BWP.

param cell_name
No help available

param padding
No help available

6.3.4.10.2.633 Periodicity

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:DL:PERiodicity
```

class PeriodicityCls

Periodicity commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:DL:PERiodicity
value: int = driver.configure.signaling.nradio.cell.ueScheduling.sps.downlink.
↪periodicity.get(cell_name = 'abc')
```

Configures the signaled 'periodicity' for DL SPS scheduling, in ms, for the initial BWP.

param cell_name
No help available

return
periodicity: No help available

set(*cell_name: str*, *periodicity: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:DL:PERiodicity
driver.configure.signaling.nradio.cell.ueScheduling.sps.downlink.periodicity.
↪set(cell_name = 'abc', periodicity = 1)
```

Configures the signaled 'periodicity' for DL SPS scheduling, in ms, for the initial BWP.

param cell_name
No help available

param periodicity
No help available

6.3.4.10.2.634 RaType

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:DL:RATYpe
```

class RaTypeCls

RaType commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → ResourceAllocationType

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:DL:RATYpe
value: enums.ResourceAllocationType = driver.configure.signaling.nradio.cell.
↪ueScheduling.sps.downlink.raType.get(cell_name = 'abc')
```

Configures the signaled 'resourceAllocation' for DL SPS scheduling, for the initial BWP.

param cell_name
type 0, type 1, dynamic switch

return
resource_allocation_type: No help available

set(*cell_name: str*, *resource_allocation_type: ResourceAllocationType*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:DL:RATYpe
driver.configure.signaling.nradio.cell.ueScheduling.sps.downlink.raType.
↪set(cell_name = 'abc', resource_allocation_type = enums.
↪ResourceAllocationType.DSWich)
```

Configures the signaled 'resourceAllocation' for DL SPS scheduling, for the initial BWP.

param cell_name
No help available

param resource_allocation_type
No help available

6.3.4.10.2.635 RbgSize

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:DL:RBGSize
```

class RbgSizeCls

RbgSize commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → RgbSize

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:DL:RBGSize
value: enums.RgbSize = driver.configure.signaling.nradio.cell.ueScheduling.sps.
↪downlink.rbgSize.get(cell_name = 'abc')
```

Configures the signaled 'rbg-Size' for DL SPS scheduling, for the initial BWP.

param cell_name
No help available

return
rgb_size: Config 1 or 2

set(*cell_name: str*, *rgb_size: RgbSize*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:DL:RBGSize
driver.configure.signaling.nradio.cell.ueScheduling.sps.downlink.rbgSize.
↪set(cell_name = 'abc', rgb_size = enums.RgbSize.CON1)
```

Configures the signaled 'rbg-Size' for DL SPS scheduling, for the initial BWP.

param cell_name
No help available

param rgb_size
Config 1 or 2

6.3.4.10.2.636 Ssid

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:DL:SSID
```

class SsidCls

Ssid commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:DL:SSID
value: int = driver.configure.signaling.nradio.cell.ueScheduling.sps.downlink.
↪ssid.get(cell_name = 'abc')
```

Configures the search space ID for DL SPS scheduling, for the initial BWP.

param cell_name
No help available

return
search_space_id: No help available

set(*cell_name: str*, *search_space_id: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UEScheduling:SPS:DL:SSID
driver.configure.signaling.nradio.cell.ueScheduling.sps.downlink.ssid.set(cell_
↪name = 'abc', search_space_id = 1)
```

Configures the search space ID for DL SPS scheduling, for the initial BWP.

param cell_name
No help available

param search_space_id
No help available

6.3.4.10.2.637 Sassignment

class SassignmentCls

Sassignment commands group definition. 14 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.ueScheduling.sps.sassignment.clone()
```

Subgroups

6.3.4.10.2.638 Downlink

class DownlinkCls

Downlink commands group definition. 7 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.ueScheduling.sps.sassignment.downlink.
↪clone()
```

Subgroups

6.3.4.10.2.639 All

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UEScheduling:SPS:SASSignment:DL:ALL
```

class AllCls

All commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Slot: int: Slot for sending the DCI that activates DL SPS.
- Number_Rb: int: No parameter help available
- Start_Rb: int: No parameter help available
- Mcs: int: No parameter help available
- Start_Symbol: int: No parameter help available
- Number_Symbol: int: No parameter help available
- Mapping: enums.Mapping: Type of PDSCH mapping
- Offset: int: Slot offset k0 for the PDSCH

class SetStruct

Structure for setting input parameters. Contains optional setting parameters. Fields:

- Cell_Name: str: No parameter help available
- Slot: int: Optional setting parameter. Slot for sending the DCI that activates DL SPS.
- Number_Rb: int: No parameter help available
- Start_Rb: int: No parameter help available
- Mcs: int: No parameter help available
- Start_Symbol: int: No parameter help available
- Number_Symbol: int: No parameter help available
- Mapping: enums.Mapping: Optional setting parameter. Type of PDSCH mapping
- Offset: int: Optional setting parameter. Slot offset k0 for the PDSCH

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:SASSignment:DL:ALL
value: GetStruct = driver.configure.signaling.nradio.cell.ueScheduling.sps.
↪sassignment.downlink.all.get(cell_name = 'abc')
```

Configures several settings for SPS DL scheduling (combination of the other ...SPS:SASSignment:DL:... commands) , for the initial BWP.

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*structure: SetStruct*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:SASSignment:DL:ALL
structure = driver.configure.signaling.nradio.cell.ueScheduling.sps.sassignment.
↪downlink.all.SetStruct()
structure.Cell_Name: str = 'abc'
```

(continues on next page)

(continued from previous page)

```
structure.Slot: int = 1
structure.Number_Rb: int = 1
structure.Start_Rb: int = 1
structure.Mcs: int = 1
structure.Start_Symbol: int = 1
structure.Number_Symbol: int = 1
structure.Mapping: enums.Mapping = enums.Mapping.A
structure.Offset: int = 1
driver.configure.signaling.nradio.cell.ueScheduling.sps.sassignment.downlink.
↪all.set(structure)
```

Configures several settings for SPS DL scheduling (combination of the other ...SPS:SASSignment:DL:... commands) , for the initial BWP.

param structure
for set value, see the help for SetStruct structure arguments.

6.3.4.10.2.640 Mcs

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UEScheduling:SPS:SASSignment:DL:MCS
```

class McsCls

Mcs commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UEScheduling:SPS:SASSignment:DL:MCS
value: int = driver.configure.signaling.nradio.cell.ueScheduling.sps.
↪sassignment.downlink.mcs.get(cell_name = 'abc')
```

Specifies the MCS index, for DL SPS scheduling, for the initial BWP.

param cell_name
No help available

return
mcs: No help available

set(*cell_name: str*, *mcs: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UEScheduling:SPS:SASSignment:DL:MCS
driver.configure.signaling.nradio.cell.ueScheduling.sps.sassignment.downlink.
↪mcs.set(cell_name = 'abc', mcs = 1)
```

Specifies the MCS index, for DL SPS scheduling, for the initial BWP.

param cell_name
No help available

param mcs
No help available

6.3.4.10.2.641 Rb

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:SASSignment:DL:RB
```

class RbCls

Rb commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Number_Rb: int: No parameter help available
- Start_Rb: int: No parameter help available

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:SASSignment:DL:RB
value: GetStruct = driver.configure.signaling.nradio.cell.ueScheduling.sps.
↪sassignment.downlink.rb.get(cell_name = 'abc')
```

Defines the scheduled contiguous RB allocation, within the BWP, for DL SPS scheduling, for the initial BWP.

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *number_rb: int*, *start_rb: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:SASSignment:DL:RB
driver.configure.signaling.nradio.cell.ueScheduling.sps.sassignment.downlink.rb.
↪set(cell_name = 'abc', number_rb = 1, start_rb = 1)
```

Defines the scheduled contiguous RB allocation, within the BWP, for DL SPS scheduling, for the initial BWP.

param cell_name
No help available

param number_rb
No help available

param start_rb
No help available

6.3.4.10.2.642 Sindex

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:SASSignment:DL:SINDex
```

class SindexCls

Sindex commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:SASSignment:DL:SINDex
value: int = driver.configure.signaling.nradio.cell.ueScheduling.sps.
↪sassignment.downlink.sindex.get(cell_name = 'abc')
```

Selects a slot for sending the DCI that activates DL SPS and informs the UE about the scheduling, for the initial BWP.

param cell_name
No help available

return
slot: No help available

set(*cell_name: str*, *slot: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:SASSignment:DL:SINDex
driver.configure.signaling.nradio.cell.ueScheduling.sps.sassignment.downlink.
↪sindex.set(cell_name = 'abc', slot = 1)
```

Selects a slot for sending the DCI that activates DL SPS and informs the UE about the scheduling, for the initial BWP.

param cell_name
No help available

param slot
No help available

6.3.4.10.2.643 Tdomain

class TdomainCls

Tdomain commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.ueScheduling.sps.sassignment.downlink.
↪tdomain.clone()
```

Subgroups

6.3.4.10.2.644 Chmapping

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:SASSignment:DL:TDOMain:CHMapping
```

class ChmappingCls

Chmapping commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → Mapping

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:SASSignment:DL:TDOMain:CHMapping
value: enums.Mapping = driver.configure.signaling.nradio.cell.ueScheduling.sps.
↪sassignment.downlink.tdomain.chmapping.get(cell_name = 'abc')
```

Selects the type of PDSCH mapping, for DL SPS scheduling, for the initial BWP.

param cell_name
No help available

return
mapping: No help available

set(*cell_name: str*, *mapping: Mapping*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:SASSignment:DL:TDOMain:CHMapping
driver.configure.signaling.nradio.cell.ueScheduling.sps.sassignment.downlink.
↪tdomain.chmapping.set(cell_name = 'abc', mapping = enums.Mapping.A)
```

Selects the type of PDSCH mapping, for DL SPS scheduling, for the initial BWP.

param cell_name
No help available

param mapping
No help available

6.3.4.10.2.645 Soffset

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:SASSignment:DL:TDOMain:SOFFset
```

class SoffsetCls

Soffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:SASSignment:DL:TDOMain:SOFFset
value: int = driver.configure.signaling.nradio.cell.ueScheduling.sps.
↪sassignment.downlink.tdomain.soffset.get(cell_name = 'abc')
```

Configures the slot offset k0 for the PDSCH, for DL SPS scheduling, for the initial BWP.

param cell_name
No help available

return
offset: No help available

set(*cell_name: str*, *offset: int*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:SASSignment:DL:TDOMain:SOFFset
driver.configure.signaling.nradio.cell.ueScheduling.sps.sassignment.downlink.
↪tdomain.soffset.set(cell_name = 'abc', offset = 1)
```

Configures the slot offset k0 for the PDSCH, for DL SPS scheduling, for the initial BWP.

param cell_name
No help available

param offset
No help available

6.3.4.10.2.646 Symbol

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:SASSignment:DL:TDOMain:SYMBol
```

class SymbolCls

Symbol commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Start_Symbol: int: No parameter help available
- Number_Symbol: int: No parameter help available

get(*cell_name: str*) → GetStruct

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:SASSignment:DL:TDOMain:SYMBol
value: GetStruct = driver.configure.signaling.nradio.cell.ueScheduling.sps.
↪sassignment.downlink.tdomain.symbol.get(cell_name = 'abc')
```

Defines the index of the first allocated OFDM symbol and the number of allocated OFDM symbols, for DL SPS scheduling, for the initial BWP.

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *start_symbol: int*, *number_symbol: int*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:SASSignment:DL:TDOMain:SYMBol
driver.configure.signaling.nradio.cell.ueScheduling.sps.sassignment.downlink.
↪tdomain.symbol.set(cell_name = 'abc', start_symbol = 1, number_symbol = 1)
```

Defines the index of the first allocated OFDM symbol and the number of allocated OFDM symbols, for DL SPS scheduling, for the initial BWP.

param cell_name
No help available

param start_symbol
No help available

param number_symbol
No help available

6.3.4.10.2.647 Uplink

class UplinkCls

Uplink commands group definition. 7 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.ueScheduling.sps.sassignment.uplink.
↪clone()
```

Subgroups

6.3.4.10.2.648 All

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:SASSignment:UL:ALL
```

class AllCls

All commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Slot: int: Slot for sending the DCI that enables UL CG.
- Number_Rb: int: No parameter help available
- Start_Rb: int: No parameter help available
- Mcs: int: No parameter help available

- Start_Symbol: int: No parameter help available
- Number_Symbol: int: No parameter help available
- Mapping: enums.Mapping: Type of PUSCH mapping
- Offset: int: Slot offset k2 for the PUSCH

class SetStruct

Structure for setting input parameters. Contains optional setting parameters. Fields:

- Cell_Name: str: No parameter help available
- Slot: int: Optional setting parameter. Slot for sending the DCI that enables UL CG.
- Number_Rb: int: No parameter help available
- Start_Rb: int: No parameter help available
- Mcs: int: No parameter help available
- Start_Symbol: int: No parameter help available
- Number_Symbol: int: No parameter help available
- Mapping: enums.Mapping: Optional setting parameter. Type of PUSCH mapping
- Offset: int: Optional setting parameter. Slot offset k2 for the PUSCH

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:SASSignment:UL:ALL
value: GetStruct = driver.configure.signaling.nradio.cell.ueScheduling.sps.
↪sassignment.uplink.all.get(cell_name = 'abc')
```

Configures several settings for UL configured grant (combination of the other ... SPS:SASSignment:UL:... commands) , for the initial BWP.

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*structure: SetStruct*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:SASSignment:UL:ALL
structure = driver.configure.signaling.nradio.cell.ueScheduling.sps.sassignment.
↪uplink.all.SetStruct()
structure.Cell_Name: str = 'abc'
structure.Slot: int = 1
structure.Number_Rb: int = 1
structure.Start_Rb: int = 1
structure.Mcs: int = 1
structure.Start_Symbol: int = 1
structure.Number_Symbol: int = 1
structure.Mapping: enums.Mapping = enums.Mapping.A
structure.Offset: int = 1
driver.configure.signaling.nradio.cell.ueScheduling.sps.sassignment.uplink.all.
↪set(structure)
```

Configures several settings for UL configured grant (combination of the other ...SPS:SASSignment:UL:... commands) , for the initial BWP.

param structure
for set value, see the help for SetStruct structure arguments.

6.3.4.10.2.649 Mcs

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:SASSignment:UL:MCS
```

class McsCls

Mcs commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:SASSignment:UL:MCS
value: int = driver.configure.signaling.nradio.cell.ueScheduling.sps.
↪sassignment.uplink.mcs.get(cell_name = 'abc')
```

Specifies the MCS index, for UL configured grant, for the initial BWP.

param cell_name
No help available

return
mcs: No help available

set(*cell_name: str*, *mcs: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:SASSignment:UL:MCS
driver.configure.signaling.nradio.cell.ueScheduling.sps.sassignment.uplink.mcs.
↪set(cell_name = 'abc', mcs = 1)
```

Specifies the MCS index, for UL configured grant, for the initial BWP.

param cell_name
No help available

param mcs
No help available

6.3.4.10.2.650 Rb

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:SASSignment:UL:RB
```

class RbCls

Rb commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Number_Rb: int: No parameter help available
- Start_Rb: int: No parameter help available

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:SASSignment:UL:RB
value: GetStruct = driver.configure.signaling.nradio.cell.ueScheduling.sps.
↪sassignment.uplink.rb.get(cell_name = 'abc')
```

Defines the scheduled contiguous RB allocation, within the BWP, for UL configured grant, for the initial BWP.

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *number_rb: int*, *start_rb: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:SASSignment:UL:RB
driver.configure.signaling.nradio.cell.ueScheduling.sps.sassignment.uplink.rb.
↪set(cell_name = 'abc', number_rb = 1, start_rb = 1)
```

Defines the scheduled contiguous RB allocation, within the BWP, for UL configured grant, for the initial BWP.

param cell_name
No help available

param number_rb
No help available

param start_rb
No help available

6.3.4.10.2.651 Sindex

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:SASSignment:UL:SINDex
```

class SindexCls

Sindex commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:SASSignment:UL:SINDex
value: int = driver.configure.signaling.nradio.cell.ueScheduling.sps.
↪sassignment.uplink.sindex.get(cell_name = 'abc')
```

Selects a slot for sending the DCI that informs the UE about the UL grant, for the initial BWP.

param cell_name
No help available

return
slot: No help available

set(*cell_name: str*, *slot: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:SASSignment:UL:SINDex
driver.configure.signaling.nradio.cell.ueScheduling.sps.sassignment.uplink.
↪sindex.set(cell_name = 'abc', slot = 1)
```

Selects a slot for sending the DCI that informs the UE about the UL grant, for the initial BWP.

param cell_name
No help available

param slot
No help available

6.3.4.10.2.652 Tdomain

class TdomainCls

Tdomain commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.ueScheduling.sps.sassignment.uplink.
↪tdomain.clone()
```

Subgroups

6.3.4.10.2.653 Chmapping

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:SASSignment:UL:TDOMain:CHMapping
```

class ChmappingCls

Chmapping commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → Mapping

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:SASSignment:UL:TDOMain:CHMapping
value: enums.Mapping = driver.configure.signaling.nradio.cell.ueScheduling.sps.
↪sassignment.uplink.tdomain.chmapping.get(cell_name = 'abc')
```

Selects the type of PUSCH mapping, for UL configured grant, for the initial BWP.

param cell_name
No help available

return
mapping: No help available

set(*cell_name: str*, *mapping: Mapping*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:SASSignment:UL:TDOMain:CHMapping
driver.configure.signaling.nradio.cell.ueScheduling.sps.sassignment.uplink.
↪tdomain.chmapping.set(cell_name = 'abc', mapping = enums.Mapping.A)
```

Selects the type of PUSCH mapping, for UL configured grant, for the initial BWP.

param cell_name
No help available

param mapping
No help available

6.3.4.10.2.654 Soffset

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:SASSignment:UL:TDOMain:SOFFset
```

class SoffsetCls

Soffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:SASSignment:UL:TDOMain:SOFFset
value: int = driver.configure.signaling.nradio.cell.ueScheduling.sps.
↪sassignment.uplink.tdomain.soffset.get(cell_name = 'abc')
```

Configures the slot offset k2 for the PUSCH, for UL configured grant, for the initial BWP.

param cell_name
No help available

return
offset: No help available

set(*cell_name: str*, *offset: int*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:SASSignment:UL:TDOMain:SOFFset
driver.configure.signaling.nradio.cell.ueScheduling.sps.sassignment.uplink.
↪tdomain.soffset.set(cell_name = 'abc', offset = 1)
```

Configures the slot offset k2 for the PUSCH, for UL configured grant, for the initial BWP.

param cell_name
No help available

param offset
No help available

6.3.4.10.2.655 Symbol

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:SASSignment:UL:TDOMain:SYMBol
```

class SymbolCls

Symbol commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Start_Symbol: int: No parameter help available
- Number_Symbol: int: No parameter help available

get(*cell_name: str*) → GetStruct

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:SASSignment:UL:TDOMain:SYMBol
value: GetStruct = driver.configure.signaling.nradio.cell.ueScheduling.sps.
↪sassignment.uplink.tdomain.symbol.get(cell_name = 'abc')
```

Defines the index of the first allocated OFDM symbol and the number of allocated OFDM symbols, for UL configured grant, for the initial BWP.

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *start_symbol: int*, *number_symbol: int*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:SASSignment:UL:TDOMain:SYMBol
driver.configure.signaling.nradio.cell.ueScheduling.sps.sassignment.uplink.
↪tdomain.symbol.set(cell_name = 'abc', start_symbol = 1, number_symbol = 1)
```

Defines the index of the first allocated OFDM symbol and the number of allocated OFDM symbols, for UL configured grant, for the initial BWP.

param cell_name
No help available

param start_symbol
No help available

param number_symbol
No help available

6.3.4.10.2.656 Uplink

class UplinkCls

Uplink commands group definition. 12 total commands, 12 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.ueScheduling.sps.uplink.clone()
```

Subgroups

6.3.4.10.2.657 Alevel

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:UL:ALEVel
```

class AlevelCls

Alevel commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → Level

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:UL:ALEVel
value: enums.Level = driver.configure.signaling.nradio.cell.ueScheduling.sps.
↪uplink.alevel.get(cell_name = 'abc')
```

Configures the aggregation level for UL configured grant, for the initial BWP.

param cell_name
No help available

return
alevel: No help available

set(*cell_name: str*, *alevel: Level*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:UL:ALEVel
driver.configure.signaling.nradio.cell.ueScheduling.sps.uplink.alevel.set(cell_
↪name = 'abc', alevel = enums.Level.AL1)
```

Configures the aggregation level for UL configured grant, for the initial BWP.

param cell_name
No help available

param alevel
No help available

6.3.4.10.2.658 All

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UEScheduling:SPS:UL:ALL
```

class AllCls

All commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Periodicity: enums.SpsPeriodicity: No parameter help available
- Mcs_Table: enums.McsTableB: 256QAM, 64QAM low SE, 64QAM
- Alevel: enums.Level: Aggregation level
- Search_Space_Id: int: No parameter help available
- Resource_Allocation_Type: enums.ResourceAllocationType: No parameter help available
- Rgb_Size: enums.RgbSize: Config 1 or 2
- No_Harq: int: Signaled 'nrofHARQ-Processes'
- Enable_Tp: bool: Signaled 'transformPrecoder'
- Timer: int: Signaled 'configuredGrantTimer'
- Rep_K: enums.PdcchFormatB: No parameter help available
- Rep_Kr_V: enums.Spreset: No parameter help available
- Position: enums.SpsPosition: Signaled 'dmrs-AdditionalPosition'

class SetStruct

Structure for setting input parameters. Contains optional setting parameters. Fields:

- Cell_Name: str: No parameter help available
- Periodicity: enums.SpsPeriodicity: No parameter help available
- Mcs_Table: enums.McsTableB: Optional setting parameter. 256QAM, 64QAM low SE, 64QAM
- Alevel: enums.Level: Optional setting parameter. Aggregation level
- Search_Space_Id: int: No parameter help available
- Resource_Allocation_Type: enums.ResourceAllocationType: No parameter help available
- Rgb_Size: enums.RgbSize: Optional setting parameter. Config 1 or 2
- No_Harq: int: Optional setting parameter. Signaled 'nrofHARQ-Processes'
- Enable_Tp: bool: Optional setting parameter. Signaled 'transformPrecoder'
- Timer: int: Optional setting parameter. Signaled 'configuredGrantTimer'
- Rep_K: enums.PdcchFormatB: No parameter help available
- Rep_Kr_V: enums.Spreset: No parameter help available
- Position: enums.SpsPosition: Optional setting parameter. Signaled 'dmrs-AdditionalPosition'

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UEScheduling:SPS:UL:ALL
value: GetStruct = driver.configure.signaling.nradio.cell.ueScheduling.sps.
↪uplink.all.get(cell_name = 'abc')
```

Configures several settings for UL configured grant (combination of the other ...SPS:UL:... commands) , for the initial BWP.

param cell_name
Type 0, type 1, dynamic switch

return
structure: for return value, see the help for GetStruct structure arguments.

set(*structure: SetStruct*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UEScheduling:SPS:UL:ALL
structure = driver.configure.signaling.nradio.cell.ueScheduling.sps.uplink.all.
↪SetStruct()
structure.Cell_Name: str = 'abc'
structure.Periodicity: enums.SpsPeriodicity = enums.SpsPeriodicity.S1
structure.Mcs_Table: enums.McsTableB = enums.McsTableB.L64
structure.Alevel: enums.Level = enums.Level.AL1
structure.Search_Space_Id: int = 1
structure.Resource_Allocation_Type: enums.ResourceAllocationType = enums.
↪ResourceAllocationType.DSWich
structure.Rgb_Size: enums.RgbSize = enums.RgbSize.CON1
structure.No_Harq: int = 1
structure.Enable_Tp: bool = False
structure.Timer: int = 1
structure.Rep_K: enums.PdcchFormatB = enums.PdcchFormatB.N1
structure.Rep_Kr_V: enums.Spreset = enums.Spreset.S1
structure.Position: enums.SpsPosition = enums.SpsPosition.POS0
driver.configure.signaling.nradio.cell.ueScheduling.sps.uplink.all.
↪set(structure)
```

Configures several settings for UL configured grant (combination of the other ...SPS:UL:... commands) , for the initial BWP.

param structure
for set value, see the help for SetStruct structure arguments.

6.3.4.10.2.659 CgTimer

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UEScheduling:SPS:UL:CGTimer
```

class CgTimerCls

CgTimer commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:UL:CGTimer
value: int = driver.configure.signaling.nradio.cell.ueScheduling.sps.uplink.
↪cgTimer.get(cell_name = 'abc')
```

Configures the signaled 'configuredGrantTimer' for UL configured grant, for the initial BWP.

param cell_name
No help available

return
timer: No help available

set(*cell_name: str*, *timer: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:UL:CGTimer
driver.configure.signaling.nradio.cell.ueScheduling.sps.uplink.cgTimer.set(cell_
↪name = 'abc', timer = 1)
```

Configures the signaled 'configuredGrantTimer' for UL configured grant, for the initial BWP.

param cell_name
No help available

param timer
No help available

6.3.4.10.2.660 DmrsPosition

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:UL:DMRSposition
```

class DmrsPositionCls

DmrsPosition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → SpsPosition

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:UL:DMRSposition
value: enums.SpsPosition = driver.configure.signaling.nradio.cell.ueScheduling.
↪sps.uplink.dmrsPosition.get(cell_name = 'abc')
```

Configures the signaled 'dmrs-AdditionalPosition' for UL configured grant, for the initial BWP.

param cell_name
No help available

return
position: No help available

set(*cell_name: str*, *position: SpsPosition*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:UL:DMRSposition
driver.configure.signaling.nradio.cell.ueScheduling.sps.uplink.dmrsPosition.
↪set(cell_name = 'abc', position = enums.SpsPosition.POS0)
```

Configures the signaled 'dmrs-AdditionalPosition' for UL configured grant, for the initial BWP.

param cell_name
No help available

param position
No help available

6.3.4.10.2.661 McsTable

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:UL:MCSTable
```

class McsTableCls

McsTable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → McsTableB

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:UL:MCSTable
value: enums.McsTableB = driver.configure.signaling.nradio.cell.ueScheduling.
↪sps.uplink.mcsTable.get(cell_name = 'abc')
```

Configures the signaled mcs-Table for UL configured grant, for the initial BWP.

param cell_name
No help available

return
mcs_table: 256QAM, 64QAM low SE, 64QAM

set(*cell_name: str*, *mcs_table: McsTableB*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:UL:MCSTable
driver.configure.signaling.nradio.cell.ueScheduling.sps.uplink.mcsTable.
↪set(cell_name = 'abc', mcs_table = enums.McsTableB.L64)
```

Configures the signaled mcs-Table for UL configured grant, for the initial BWP.

param cell_name
No help available

param mcs_table
256QAM, 64QAM low SE, 64QAM

6.3.4.10.2.662 Nohp

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:UL:NOHP
```

class NohpCls

Nohp commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UEScheduling:SPS:UL:NOHP
value: int = driver.configure.signaling.nradio.cell.ueScheduling.sps.uplink.
↪nohp.get(cell_name = 'abc')
```

Configures the signaled 'nrofHARQ-Processes' for UL configured grant, for the initial BWP.

param cell_name
No help available

return
no_harq: No help available

set(*cell_name: str*, *no_harq: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UEScheduling:SPS:UL:NOHP
driver.configure.signaling.nradio.cell.ueScheduling.sps.uplink.nohp.set(cell_
↪name = 'abc', no_harq = 1)
```

Configures the signaled 'nrofHARQ-Processes' for UL configured grant, for the initial BWP.

param cell_name
No help available

param no_harq
No help available

6.3.4.10.2.663 Periodicity

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UEScheduling:SPS:UL:PERiodicity
```

class PeriodicityCls

Periodicity commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → SpsPeriodicity

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UEScheduling:SPS:UL:PERiodicity
value: enums.SpsPeriodicity = driver.configure.signaling.nradio.cell.
↪ueScheduling.sps.uplink.periodicity.get(cell_name = 'abc')
```

Configures the signaled 'periodicity' for UL configured grant, for the initial BWP.

param cell_name
No help available

return
periodicity: SYMn: n symbols Sn: n slots S1K, S1K2, S2K, S5K: 1024, 1280, 2560, 5120 slots

set(*cell_name: str*, *periodicity: SpsPeriodicity*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UEScheduling:SPS:UL:PERiodicity
driver.configure.signaling.nradio.cell.ueScheduling.sps.uplink.periodicity.
↪set(cell_name = 'abc', periodicity = enums.SpsPeriodicity.S1)
```

Configures the signaled 'periodicity' for UL configured grant, for the initial BWP.

param cell_name
No help available

param periodicity
SYMn: n symbols Sn: n slots S1K, S1K2, S2K, S5K: 1024, 1280, 2560, 5120 slots

6.3.4.10.2.664 RaType

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:UL:RATYpe
```

class RaTypeCls

RaType commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → ResourceAllocationType

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:UL:RATYpe
value: enums.ResourceAllocationType = driver.configure.signaling.nradio.cell.
↪ueScheduling.sps.uplink.raType.get(cell_name = 'abc')
```

Configures the signaled 'resourceAllocation' for UL configured grant, for the initial BWP.

param cell_name
type 0, type 1, dynamic switch

return
resource_allocation_type: No help available

set(*cell_name: str*, *resource_allocation_type: ResourceAllocationType*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:UL:RATYpe
driver.configure.signaling.nradio.cell.ueScheduling.sps.uplink.raType.set(cell_
↪name = 'abc', resource_allocation_type = enums.ResourceAllocationType.DSWich)
```

Configures the signaled 'resourceAllocation' for UL configured grant, for the initial BWP.

param cell_name
No help available

param resource_allocation_type
No help available

6.3.4.10.2.665 RbgSize

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:UL:RBGSize
```

class RbgSizeCls

RbgSize commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → RgbSize

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:UL:RBGSize
value: enums.RgbSize = driver.configure.signaling.nradio.cell.ueScheduling.sps.
↪uplink.rbgSize.get(cell_name = 'abc')
```

Configures the signaled 'rbg-Size' for UL configured grant, for the initial BWP.

param cell_name
No help available

return
rgb_size: Config 1 or 2

set(*cell_name: str*, *rgb_size: RgbSize*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:UL:RBGSize
driver.configure.signaling.nradio.cell.ueScheduling.sps.uplink.rbgSize.set(cell_
↪name = 'abc', rgb_size = enums.RgbSize.CON1)
```

Configures the signaled 'rbg-Size' for UL configured grant, for the initial BWP.

param cell_name
No help available

param rgb_size
Config 1 or 2

6.3.4.10.2.666 RrVersion

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:UL:RRVersion
```

class RrVersionCls

RrVersion commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Rep_K: enums.PdcchFormatB: No parameter help available
- Rep_Kr_V: enums.Spreset: No parameter help available

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:UL:RRVersion
value: GetStruct = driver.configure.signaling.nradio.cell.ueScheduling.sps.
↪uplink.rrVersion.get(cell_name = 'abc')
```

Configures the signaled 'repK' and 'repK-RV' for UL configured grant, for the initial BWP.

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *rep_k: PdcchFormatB*, *rep_kr_v: Spreset*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:UL:RRVersion
driver.configure.signaling.nradio.cell.ueScheduling.sps.uplink.rrVersion.
↪set(cell_name = 'abc', rep_k = enums.PdcchFormatB.N1, rep_kr_v = enums.
↪Spreset.S1)
```

Configures the signaled 'repK' and 'repK-RV' for UL configured grant, for the initial BWP.

param cell_name
No help available

param rep_k
No help available

param rep_kr_v
No help available

6.3.4.10.2.667 Ssid

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:UL:SSID
```

class SsidCls

Ssid commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:UL:SSID
value: int = driver.configure.signaling.nradio.cell.ueScheduling.sps.uplink.
↪ssid.get(cell_name = 'abc')
```

Configures the search space ID for UL configured grant, for the initial BWP.

param cell_name
No help available

return
search_space_id: No help available

set(*cell_name: str*, *search_space_id: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:UL:SSID
driver.configure.signaling.nradio.cell.ueScheduling.sps.uplink.ssid.set(cell_
↪name = 'abc', search_space_id = 1)
```

Configures the search space ID for UL configured grant, for the initial BWP.

param cell_name
No help available

param search_space_id
No help available

6.3.4.10.2.668 TpEnable

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:UL:TPENable
```

class TpEnableCls

TpEnable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:UL:TPENable
value: bool = driver.configure.signaling.nradio.cell.ueScheduling.sps.uplink.
↪tpEnable.get(cell_name = 'abc')
```

Configures the signaled 'transformPrecoder' for UL configured grant, for the initial BWP.

param cell_name
No help available

return
enable_tp: ON: DFT-s-OFDM (with transform precoding) . OFF: CP-OFDM (no transform precoding) .

set(*cell_name: str*, *enable_tp: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:SPS:UL:TPENable
driver.configure.signaling.nradio.cell.ueScheduling.sps.uplink.tpEnable.
↪set(cell_name = 'abc', enable_tp = False)
```

Configures the signaled 'transformPrecoder' for UL configured grant, for the initial BWP.

param cell_name
No help available

param enable_tp
ON: DFT-s-OFDM (with transform precoding) . OFF: CP-OFDM (no transform precoding) .

6.3.4.10.2.669 UserDefined

class UserDefinedCls

UserDefined commands group definition. 36 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.ueScheduling.userDefined.clone()
```

Subgroups

6.3.4.10.2.670 CsScheduling

class CsSchedulingCls

CsScheduling commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.ueScheduling.userDefined.csScheduling.
↪clone()
```

Subgroups

6.3.4.10.2.671 K0

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:CSSCheduling:K0
```

class K0Cls

K0 commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:CSSCheduling:K0
value: int or bool = driver.configure.signaling.nradio.cell.ueScheduling.
↪userDefined.csScheduling.k0.get(cell_name = 'abc')
```

Sends a slot offset as 'minimumSchedulingOffsetK0-r16' to the UE, for the initial BWP. The slot offset defines the minimum allowed k0 value (offset between PDCCH and PDSCH, cross-slot scheduling) .

param cell_name
No help available

return
k_0: (integer or boolean) integer: Send this value. OFF: Do not send a value.

set(*cell_name: str*, *k_0: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:CSSCheduling:K0
driver.configure.signaling.nradio.cell.ueScheduling.userDefined.csScheduling.k0.
↪set(cell_name = 'abc', k_0 = 1)
```

Sends a slot offset as 'minimumSchedulingOffsetK0-r16' to the UE, for the initial BWP. The slot offset defines the minimum allowed k0 value (offset between PDCCH and PDSCH, cross-slot scheduling) .

param cell_name
No help available

param k_0
(integer or boolean) integer: Send this value. OFF: Do not send a value.

6.3.4.10.2.672 K2

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UEScheduling:UDEFined:CSScheduling:K2
```

class K2Cls

K2 commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UEScheduling:UDEFined:CSScheduling:K2
value: int or bool = driver.configure.signaling.nradio.cell.ueScheduling.
↪userDefined.csScheduling.k2.get(cell_name = 'abc')
```

Sends a slot offset as 'minimumSchedulingOffsetK2-r16' to the UE, for the initial BWP. The slot offset defines the minimum allowed k2 value (offset between PDCCH and PUSCH, cross-slot scheduling) .

param cell_name
No help available

return
k_2: (integer or boolean) integer: Send this value. OFF: Do not send a value.

set(*cell_name: str*, *k_2: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UEScheduling:UDEFined:CSScheduling:K2
driver.configure.signaling.nradio.cell.ueScheduling.userDefined.csScheduling.k2.
↪set(cell_name = 'abc', k_2 = 1)
```

Sends a slot offset as 'minimumSchedulingOffsetK2-r16' to the UE, for the initial BWP. The slot offset defines the minimum allowed k2 value (offset between PDCCH and PUSCH, cross-slot scheduling) .

param cell_name
No help available

param k_2
(integer or boolean) integer: Send this value. OFF: Do not send a value.

6.3.4.10.2.673 Downlink

class DownlinkCls

Downlink commands group definition. 6 total commands, 6 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.ueScheduling.userDefined.downlink.clone()
```

Subgroups

6.3.4.10.2.674 Alevel

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:DL:ALEVel
```

class AlevelCls

Alevel commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → Level

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:DL:ALEVel
value: enums.Level = driver.configure.signaling.nradio.cell.ueScheduling.
↪userDefined.downlink.alevel.get(cell_name = 'abc')
```

Specifies the aggregation level for the DL, for the initial BWP.

param cell_name
No help available

return
level: No help available

set(*cell_name: str*, *level: Level*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:DL:ALEVel
driver.configure.signaling.nradio.cell.ueScheduling.userDefined.downlink.alevel.
↪set(cell_name = 'abc', level = enums.Level.AL1)
```

Specifies the aggregation level for the DL, for the initial BWP.

param cell_name
No help available

param level
No help available

6.3.4.10.2.675 Bpid

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:DL:BPID
```

class BpidCls

Bpid commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:DL:BPID
value: int = driver.configure.signaling.nradio.cell.ueScheduling.userDefined.
↪downlink.bpid.get(cell_name = 'abc')
```

Queries the ID of the DL bandwidth part that contains the scheduled allocation, for the initial BWP.

param cell_name
No help available

return
idn: No help available

6.3.4.10.2.676 McsTable

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:DL:MCSTable
```

class McsTableCls

McsTable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → McsTableB

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:DL:MCSTable
value: enums.McsTableB = driver.configure.signaling.nradio.cell.ueScheduling.
↪userDefined.downlink.mcsTable.get(cell_name = 'abc')
```

Selects the MCS table that the UE must use to determine the modulation order and the target code rate of the PDSCH, for the initial BWP.

param cell_name
No help available

return
mcs_table: 256QAM, 64QAM low SE, 64QAM

set(*cell_name: str*, *mcs_table: McsTableB*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:DL:MCSTable
driver.configure.signaling.nradio.cell.ueScheduling.userDefined.downlink.
↪mcsTable.set(cell_name = 'abc', mcs_table = enums.McsTableB.L64)
```

Selects the MCS table that the UE must use to determine the modulation order and the target code rate of the PDSCH, for the initial BWP.

param cell_name
No help available

param mcs_table
256QAM, 64QAM low SE, 64QAM

6.3.4.10.2.677 Padding

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:DL:PADDing
```

class PaddingCls

Padding commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:DL:PADDing
value: bool = driver.configure.signaling.nradio.cell.ueScheduling.userDefined.
↪downlink.padding.get(cell_name = 'abc')
```

Activates or deactivates downlink padding at the MAC layer, for the initial BWP.

param cell_name
No help available

return
enable: No help available

set(*cell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:DL:PADDing
driver.configure.signaling.nradio.cell.ueScheduling.userDefined.downlink.
↪padding.set(cell_name = 'abc', enable = False)
```

Activates or deactivates downlink padding at the MAC layer, for the initial BWP.

param cell_name
No help available

param enable
No help available

6.3.4.10.2.678 Ssid

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:DL:SSID
```

class SsidCls

Ssid commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:DL:SSID
value: int = driver.configure.signaling.nradio.cell.ueScheduling.userDefined.
↪downlink.ssid.get(cell_name = 'abc')
```

Queries the ID of the search space for the DL, for the initial BWP.

param cell_name
No help available

return
idn: No help available

6.3.4.10.2.679 VpMapping

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:DL:VPMapping
```

class VpMappingCls

VpMapping commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → MappingI

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:DL:VPMapping
value: enums.MappingI = driver.configure.signaling.nradio.cell.ueScheduling.
↪userDefined.downlink.vpMapping.get(cell_name = 'abc')
```

Selects whether interleaved or non-interleaved virtual RB to physical RB mapping is applied to the PDSCH, for the initial BWP.

param cell_name
No help available

return
mapping: Interleaved or non-interleaved

set(*cell_name: str*, *mapping: MappingI*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:DL:VPMapping
driver.configure.signaling.nradio.cell.ueScheduling.userDefined.downlink.
↪vpMapping.set(cell_name = 'abc', mapping = enums.MappingI.INT)
```

Selects whether interleaved or non-interleaved virtual RB to physical RB mapping is applied to the PDSCH, for the initial BWP.

param cell_name
No help available

param mapping
Interleaved or non-interleaved

6.3.4.10.2.680 Sassignment

class SassignmentCls

Sassignment commands group definition. 21 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.ueScheduling.userDefined.sassignment.
↪clone()
```

Subgroups

6.3.4.10.2.681 Downlink

class DownlinkCls

Downlink commands group definition. 10 total commands, 7 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.ueScheduling.userDefined.sassignment.
↪downlink.clone()
```

Subgroups

6.3.4.10.2.682 All

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:DL:ALL
```

class AllCls

All commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Slot: List[int]: Index number of the slot
- Enable: List[bool]: No parameter help available
- Number_Rb: List[int]: No parameter help available
- Start_Rb: List[int]: No parameter help available
- Mcs: List[int]: No parameter help available
- Dci_Format: List[enums.DciFormatB]: No parameter help available
- Mimo: List[enums.Mimo]: No parameter help available

class SetStruct

Structure for setting input parameters. Fields:

- Cell_Name: str: No parameter help available
- Slot: List[int]: Index number of the slot
- Enable: List[bool]: No parameter help available
- Number_Rb: List[int]: No parameter help available
- Start_Rb: List[int]: No parameter help available
- Mcs: List[int]: No parameter help available
- Dci_Format: List[enums.DciFormatB]: No parameter help available

- Mimo: List[enums.Mimo]: No parameter help available

get(*cell_name: str*) → GetStruct

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:DL:ALL
value: GetStruct = driver.configure.signaling.nradio.cell.ueScheduling.
↪userDefined.sassignment.downlink.all.get(cell_name = 'abc')
```

Defines scheduling settings for one or more DL slots, for the initial BWP. The parameter sequence contains one set of values per slot: <CellName>, {<Slot>, <Enable>, <NumberRB>, <StartRB>, <MCS>, <DCIFormat>, <Mimo>}slot a, {...}slot b, ... A query returns all DL slots.

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*structure: SetStruct*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:DL:ALL
structure = driver.configure.signaling.nradio.cell.ueScheduling.userDefined.
↪sassignment.downlink.all.SetStruct()
structure.Cell_Name: str = 'abc'
structure.Slot: List[int] = [1, 2, 3]
structure.Enable: List[bool] = [True, False, True]
structure.Number_Rb: List[int] = [1, 2, 3]
structure.Start_Rb: List[int] = [1, 2, 3]
structure.Mcs: List[int] = [1, 2, 3]
structure.Dci_Format: List[enums.DciFormatB] = [DciFormatB.D10, DciFormatB.D11]
structure.Mimo: List[enums.Mimo] = [Mimo.M22, Mimo.SISO]
driver.configure.signaling.nradio.cell.ueScheduling.userDefined.sassignment.
↪downlink.all.set(structure)
```

Defines scheduling settings for one or more DL slots, for the initial BWP. The parameter sequence contains one set of values per slot: <CellName>, {<Slot>, <Enable>, <NumberRB>, <StartRB>, <MCS>, <DCIFormat>, <Mimo>}slot a, {...}slot b, ... A query returns all DL slots.

param structure
for set value, see the help for SetStruct structure arguments.

6.3.4.10.2.683 DciFormat

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:DL:DCIFormat
```

class DciFormatCls

DciFormat commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *slot: int*) → DciFormatB

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:DL:DCIFormat
value: enums.DciFormatB = driver.configure.signaling.nradio.cell.ueScheduling.
↪userDefined.sassignment.downlink.dciFormat.get(cell_name = 'abc', slot = 1)
```

Defines the DCI format for the DL slot with the index <Slot>, for the initial BWP.

param cell_name
No help available

param slot
No help available

return
dci_format: No help available

set(*cell_name: str*, *slot: int*, *dci_format: DciFormatB*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:DL:DCIFormat
driver.configure.signaling.nradio.cell.ueScheduling.userDefined.sassignment.
↪downlink.dciFormat.set(cell_name = 'abc', slot = 1, dci_format = enums.
↪DciFormatB.D10)
```

Defines the DCI format for the DL slot with the index <Slot>, for the initial BWP.

param cell_name
No help available

param slot
No help available

param dci_format
No help available

6.3.4.10.2.684 Enable

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:DL:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *slot: int*) → bool

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:DL:ENABle
value: bool = driver.configure.signaling.nradio.cell.ueScheduling.userDefined.
↪sassignment.downlink.enable.get(cell_name = 'abc', slot = 1)
```

Enables or disables scheduling of the DL slot with the index <Slot>, for the initial BWP.

param cell_name
No help available

param slot
No help available

return
enable: No help available

set(*cell_name: str*, *slot: int*, *enable: bool*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:DL:ENABle
driver.configure.signaling.nradio.cell.ueScheduling.userDefined.sassignment.
↪downlink.enable.set(cell_name = 'abc', slot = 1, enable = False)
```

Enables or disables scheduling of the DL slot with the index <Slot>, for the initial BWP.

param cell_name
No help available

param slot
No help available

param enable
No help available

6.3.4.10.2.685 Mcs

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:DL:MCS
```

class McsCls

Mcs commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *slot: int*) → int

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:DL:MCS
value: int = driver.configure.signaling.nradio.cell.ueScheduling.userDefined.
↪sassignment.downlink.mcs.get(cell_name = 'abc', slot = 1)
```

Specifies the MCS index for the DL slot with the index <Slot>, for the initial BWP.

param cell_name
No help available

param slot
No help available

return
mcs: No help available

set(*cell_name: str*, *slot: int*, *mcs: int*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:DL:MCS
driver.configure.signaling.nradio.cell.ueScheduling.userDefined.sassignment.
↪downlink.mcs.set(cell_name = 'abc', slot = 1, mcs = 1)
```

Specifies the MCS index for the DL slot with the index <Slot>, for the initial BWP.

param cell_name
No help available

param slot
No help available

param mcs
No help available

6.3.4.10.2.686 Mimo

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:DL:MIMO
```

class MimoCls

Mimo commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *slot: int*) → Mimo

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:DL:MIMO
value: enums.Mimo = driver.configure.signaling.nradio.cell.ueScheduling.
↪userDefined.sassignment.downlink.mimo.get(cell_name = 'abc', slot = 1)
```

Specifies the MIMO scheme for the DL slot with the index <Slot>, for the initial BWP.

param cell_name
No help available

param slot
No help available

return
mimo: SISO: 1xN M22: 2xN M33: 3xN M44: 4xN

set(*cell_name: str*, *slot: int*, *mimo: Mimo*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:DL:MIMO
driver.configure.signaling.nradio.cell.ueScheduling.userDefined.sassignment.
↪downlink.mimo.set(cell_name = 'abc', slot = 1, mimo = enums.Mimo.M22)
```

Specifies the MIMO scheme for the DL slot with the index <Slot>, for the initial BWP.

param cell_name
No help available

param slot
No help available

param mimo
SISO: 1xN M22: 2xN M33: 3xN M44: 4xN

6.3.4.10.2.687 Rb

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:DL:RB
```

class RbCls

Rb commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Number_Rb: int: No parameter help available
- Start_Rb: int: No parameter help available

get(*cell_name: str*, *slot: int*) → GetStruct

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:DL:RB
value: GetStruct = driver.configure.signaling.nradio.cell.ueScheduling.
↪userDefined.sassignment.downlink.rb.get(cell_name = 'abc', slot = 1)
```

Specifies the scheduled RB allocation for the DL slot with the index <Slot>, for the initial BWP.

param cell_name
No help available

param slot
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *slot: int*, *number_rb: int*, *start_rb: int*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:DL:RB
driver.configure.signaling.nradio.cell.ueScheduling.userDefined.sassignment.
↪downlink.rb.set(cell_name = 'abc', slot = 1, number_rb = 1, start_rb = 1)
```

Specifies the scheduled RB allocation for the DL slot with the index <Slot>, for the initial BWP.

param cell_name
No help available

param slot
No help available

param number_rb
No help available

param start_rb
No help available

6.3.4.10.2.688 Tdomain

class TdomainCls

Tdomain commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.ueScheduling.userDefined.sassignment.
↪downlink.tdomain.clone()
```

Subgroups

6.3.4.10.2.689 AnsOffset

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:DL:TDOMain:ANSoffset
```

class AnsOffsetCls

AnsOffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *slot: int*) → int

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:DL:TDOMain:ANSoffset
value: int = driver.configure.signaling.nradio.cell.ueScheduling.userDefined.
↪sassignment.downlink.tdomain.ansOffset.get(cell_name = 'abc', slot = 1)
```

Configures the ACK/NACK slot offset k1, for the DL slot with the index <Slot>, for the initial BWP.

param cell_name
No help available

param slot
No help available

return
offset: No help available

set(*cell_name: str*, *slot: int*, *offset: int*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:DL:TDOMain:ANSoffset
driver.configure.signaling.nradio.cell.ueScheduling.userDefined.sassignment.
↪downlink.tdomain.ansOffset.set(cell_name = 'abc', slot = 1, offset = 1)
```

Configures the ACK/NACK slot offset k1, for the DL slot with the index <Slot>, for the initial BWP.

param cell_name
No help available

param slot
No help available

param offset
No help available

6.3.4.10.2.690 Chmapping

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:DL:TDOMain:CHMapping
```

class ChmappingCls

Chmapping commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *slot: int*) → Mapping

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:DL:TDOMain:CHMapping
value: enums.Mapping = driver.configure.signaling.nradio.cell.ueScheduling.
↪userDefined.sassignment.downlink.tdomain.chmapping.get(cell_name = 'abc',␣
↪slot = 1)
```

Selects the type of PDSCH mapping, for the DL slot with the index <Slot>, for the initial BWP.

param cell_name
No help available

param slot
No help available

return
mapping: No help available

set(*cell_name: str*, *slot: int*, *mapping: Mapping*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:DL:TDOMain:CHMapping
driver.configure.signaling.nradio.cell.ueScheduling.userDefined.sassignment.
↪downlink.tdomain.chmapping.set(cell_name = 'abc', slot = 1, mapping = enums.
↪Mapping.A)
```

Selects the type of PDSCH mapping, for the DL slot with the index <Slot>, for the initial BWP.

param cell_name
No help available

param slot
No help available

param mapping
No help available

6.3.4.10.2.691 Soffset

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:DL:TDOMain:SOFFset
```

class SoffsetCls

Soffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *slot: int*) → int

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:DL:TDOMain:SOFFset
value: int = driver.configure.signaling.nradio.cell.ueScheduling.userDefined.
↪sassignment.downlink.tdomain.soffset.get(cell_name = 'abc', slot = 1)
```

Configures the slot offset k0 for the PDSCH, for the DL slot with the index <Slot>, for the initial BWP.

param cell_name
No help available

param slot
No help available

return
offset: No help available

set(*cell_name: str*, *slot: int*, *offset: int*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:DL:TDOMain:SOFFset
driver.configure.signaling.nradio.cell.ueScheduling.userDefined.sassignment.
↪downlink.tdomain.soffset.set(cell_name = 'abc', slot = 1, offset = 1)
```

Configures the slot offset k0 for the PDSCH, for the DL slot with the index <Slot>, for the initial BWP.

param cell_name
No help available

param slot
No help available

param offset
No help available

6.3.4.10.2.692 Symbol

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:DL:TDOMain:SYMBol
```

class SymbolCls

Symbol commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Number_Symbol: int: No parameter help available
- Start_Symbol: int: No parameter help available

get(*cell_name: str*, *slot: int*) → GetStruct

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:DL:TDOMain:SYMBol
value: GetStruct = driver.configure.signaling.nradio.cell.ueScheduling.
↪userDefined.sassignment.downlink.tdomain.symbol.get(cell_name = 'abc', slot =␣
↪1)
```

Defines the number of allocated OFDM symbols and the index of the first allocated OFDM symbol for the PDSCH. For the DL slot with the index <Slot>, for the initial BWP.

param cell_name
No help available

param slot
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *slot: int*, *number_symbol: int*, *start_symbol: int*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:DL:TDOMain:SYMBol
driver.configure.signaling.nradio.cell.ueScheduling.userDefined.sassignment.
↪downlink.tdomain.symbol.set(cell_name = 'abc', slot = 1, number_symbol = 1,␣
↪start_symbol = 1)
```

Defines the number of allocated OFDM symbols and the index of the first allocated OFDM symbol for the PDSCH. For the DL slot with the index <Slot>, for the initial BWP.

param cell_name
No help available

param slot
No help available

param number_symbol
No help available

param start_symbol
No help available

6.3.4.10.2.693 Uplink

class UplinkCls

Uplink commands group definition. 11 total commands, 8 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.ueScheduling.userDefined.sassignment.
↪uplink.clone()
```

Subgroups

6.3.4.10.2.694 All

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:UL:ALL
```

class AllCls

All commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Slot: List[int]: Index number of the slot
- Enable: List[bool]: No parameter help available
- Number_Rb: List[int]: No parameter help available
- Start_Rb: List[int]: No parameter help available
- Mcs: List[int]: No parameter help available
- Dci_Format: List[enums.DciFormatC]: No parameter help available
- Mimo: List[enums.MimoB]: No parameter help available

class SetStruct

Structure for setting input parameters. Fields:

- Cell_Name: str: No parameter help available
- Slot: List[int]: Index number of the slot
- Enable: List[bool]: No parameter help available
- Number_Rb: List[int]: No parameter help available
- Start_Rb: List[int]: No parameter help available
- Mcs: List[int]: No parameter help available
- Dci_Format: List[enums.DciFormatC]: No parameter help available
- Mimo: List[enums.MimoB]: No parameter help available

get(*cell_name: str*) → GetStruct

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:UL:ALL
value: GetStruct = driver.configure.signaling.nradio.cell.ueScheduling.
↪userDefined.sassignment.uplink.all.get(cell_name = 'abc')
```

Defines scheduling settings for one or more UL slots, for the initial BWP. The parameter sequence contains one set of values per slot: <CellName>, {<Slot>, <Enable>, <NumberRB>, <StartRB>, <MCS>, <DCIFormat>, <Mimo>}slot a, {...}slot b, ... A query returns all UL slots.

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*structure: SetStruct*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:UL:ALL
structure = driver.configure.signaling.nradio.cell.ueScheduling.userDefined.
↪sassignment.uplink.all.SetStruct()
structure.Cell_Name: str = 'abc'
structure.Slot: List[int] = [1, 2, 3]
structure.Enable: List[bool] = [True, False, True]
structure.Number_Rb: List[int] = [1, 2, 3]
structure.Start_Rb: List[int] = [1, 2, 3]
structure.Mcs: List[int] = [1, 2, 3]
structure.Dci_Format: List[enums.DciFormatC] = [DciFormatC.D00, DciFormatC.D01]
structure.Mimo: List[enums.MimoB] = [MimoB.M22, MimoB.SISO]
driver.configure.signaling.nradio.cell.ueScheduling.userDefined.sassignment.
↪uplink.all.set(structure)
```

Defines scheduling settings for one or more UL slots, for the initial BWP. The parameter sequence contains one set of values per slot: <CellName>, {<Slot>, <Enable>, <NumberRB>, <StartRB>, <MCS>, <DCIFormat>, <Mimo>}slot a, {...}slot b, ... A query returns all UL slots.

param structure
for set value, see the help for SetStruct structure arguments.

6.3.4.10.2.695 DciFormat

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:UL:DCIFormat
```

class DciFormatCls

DciFormat commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *slot: int*) → DciFormatC

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:UL:DCIFormat
```

(continues on next page)

(continued from previous page)

```
value: enums.DciFormatC = driver.configure.signaling.nradio.cell.ueScheduling.
↪userDefined.sassignment.uplink.dciFormat.get(cell_name = 'abc', slot = 1)
```

Defines the DCI format for the UL slot with the index <Slot>, for the initial BWP.

param cell_name
No help available

param slot
No help available

return
dci_format: No help available

set(*cell_name: str*, *slot: int*, *dci_format: DciFormatC*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:UL:DCIFormat
driver.configure.signaling.nradio.cell.ueScheduling.userDefined.sassignment.
↪uplink.dciFormat.set(cell_name = 'abc', slot = 1, dci_format = enums.
↪DciFormatC.D00)
```

Defines the DCI format for the UL slot with the index <Slot>, for the initial BWP.

param cell_name
No help available

param slot
No help available

param dci_format
No help available

6.3.4.10.2.696 Enable

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:UL:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *slot: int*) → bool

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:UL:ENABle
value: bool = driver.configure.signaling.nradio.cell.ueScheduling.userDefined.
↪sassignment.uplink.enable.get(cell_name = 'abc', slot = 1)
```

Enables or disables scheduling of the UL slot with the index <Slot>, for the initial BWP.

param cell_name
No help available

param slot
No help available

return
enable: No help available

set(*cell_name: str*, *slot: int*, *enable: bool*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:UL:ENABle
driver.configure.signaling.nradio.cell.ueScheduling.userDefined.sassignment.
↪uplink.enable.set(cell_name = 'abc', slot = 1, enable = False)
```

Enables or disables scheduling of the UL slot with the index <Slot>, for the initial BWP.

param cell_name
No help available

param slot
No help available

param enable
No help available

6.3.4.10.2.697 Mcs

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:UL:MCS
```

class McsCls

Mcs commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *slot: int*) → int

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:UL:MCS
value: int = driver.configure.signaling.nradio.cell.ueScheduling.userDefined.
↪sassignment.uplink.mcs.get(cell_name = 'abc', slot = 1)
```

Specifies the MCS index for the UL slot with the index <Slot>, for the initial BWP.

param cell_name
No help available

param slot
No help available

return
mcs: No help available

set(*cell_name: str*, *slot: int*, *mcs: int*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:UL:MCS
driver.configure.signaling.nradio.cell.ueScheduling.userDefined.sassignment.
↪uplink.mcs.set(cell_name = 'abc', slot = 1, mcs = 1)
```

Specifies the MCS index for the UL slot with the index <Slot>, for the initial BWP.

param cell_name
No help available

param slot
No help available

param mcs
No help available

6.3.4.10.2.698 Mimo

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:UL:MIMO
```

class MimoCls

Mimo commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *slot: int*) → MimoB

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:UL:MIMO
value: enums.MimoB = driver.configure.signaling.nradio.cell.ueScheduling.
↪userDefined.sassignment.uplink.mimo.get(cell_name = 'abc', slot = 1)
```

Specifies the MIMO scheme for the UL slot with the index <Slot>, for the initial BWP.

param cell_name
No help available

param slot
No help available

return
mimo: SISO: nx1 M22: nx2

set(*cell_name: str*, *slot: int*, *mimo: MimoB*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:UL:MIMO
driver.configure.signaling.nradio.cell.ueScheduling.userDefined.sassignment.
↪uplink.mimo.set(cell_name = 'abc', slot = 1, mimo = enums.MimoB.M22)
```

Specifies the MIMO scheme for the UL slot with the index <Slot>, for the initial BWP.

param cell_name
No help available

param slot
No help available

param mimo
SISO: nx1 M22: nx2

6.3.4.10.2.699 Rb

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:UL:RB
```

class RbCls

Rb commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Number_Rb: int: No parameter help available
- Start_Rb: int: No parameter help available

get(*cell_name: str*, *slot: int*) → GetStruct

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:UL:RB
value: GetStruct = driver.configure.signaling.nradio.cell.ueScheduling.
↪userDefined.sassignment.uplink.rb.get(cell_name = 'abc', slot = 1)
```

Specifies the scheduled RB allocation for the UL slot with the index <Slot>, for the initial BWP.

param cell_name
No help available

param slot
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *slot: int*, *number_rb: int*, *start_rb: int*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:UL:RB
driver.configure.signaling.nradio.cell.ueScheduling.userDefined.sassignment.
↪uplink.rb.set(cell_name = 'abc', slot = 1, number_rb = 1, start_rb = 1)
```

Specifies the scheduled RB allocation for the UL slot with the index <Slot>, for the initial BWP.

param cell_name
No help available

param slot
No help available

param number_rb
No help available

param start_rb
No help available

6.3.4.10.2.700 Tdomain

class TdomainCls

Tdomain commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.ueScheduling.userDefined.sassignment.
↪uplink.tdomain.clone()
```

Subgroups

6.3.4.10.2.701 Chmapping

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:UL:TDOMain:CHMapping
```

class ChmappingCls

Chmapping commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *slot: int*) → Mapping

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:UL:TDOMain:CHMapping
value: enums.Mapping = driver.configure.signaling.nradio.cell.ueScheduling.
↪userDefined.sassignment.uplink.tdomain.chmapping.get(cell_name = 'abc', slot␣
↪= 1)
```

Selects the type of PUSCH mapping, for the UL slot with the index <Slot>, for the initial BWP.

param cell_name
No help available

param slot
No help available

return
mapping: No help available

set(*cell_name: str*, *slot: int*, *mapping: Mapping*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:UL:TDOMain:CHMapping
driver.configure.signaling.nradio.cell.ueScheduling.userDefined.sassignment.
↪uplink.tdomain.chmapping.set(cell_name = 'abc', slot = 1, mapping = enums.
↪Mapping.A)
```

Selects the type of PUSCH mapping, for the UL slot with the index <Slot>, for the initial BWP.

param cell_name
No help available

param slot
No help available

param mapping
No help available

6.3.4.10.2.702 PnoRepet

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:UL:TDOMain:PNORepet
```

class PnoRepetCls

PnoRepet commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *slot: int*) → Repetitions

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:UL:TDOMain:PNORepet
value: enums.Repetitions = driver.configure.signaling.nradio.cell.ueScheduling.
↪userDefined.sassignment.uplink.tdomain.pnoRepet.get(cell_name = 'abc', slot =␣
↪1)
```

Specifies the number of PUSCH repetitions signaled as 'numberOfRepetitions', for the UL slot with the index <Slot>, for the initial BWP. Prerequisite: Configure [CONFigure:]SIGNaling:NRADio:CELL:UESCheduling:UDEFined:UL:PRTYpe.

param cell_name
No help available

param slot
No help available

return
repetitions: No help available

set(*cell_name: str*, *slot: int*, *repetitions: Repetitions*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:UL:TDOMain:PNORepet
driver.configure.signaling.nradio.cell.ueScheduling.userDefined.sassignment.
↪uplink.tdomain.pnoRepet.set(cell_name = 'abc', slot = 1, repetitions = enums.
↪Repetitions.N12)
```

Specifies the number of PUSCH repetitions signaled as 'numberOfRepetitions', for the UL slot with the index <Slot>, for the initial BWP. Prerequisite: Configure [CONFigure:]SIGNaling:NRADio:CELL:UESCheduling:UDEFined:UL:PRTYpe.

param cell_name
No help available

param slot
No help available

param repetitions
No help available

6.3.4.10.2.703 Soffset

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:UL:TDOMain:SOFFset
```

class SoffsetCls

Soffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *slot: int*) → int

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:UL:TDOMain:SOFFset
value: int = driver.configure.signaling.nradio.cell.ueScheduling.userDefined.
↪sassignment.uplink.tdomain.soffset.get(cell_name = 'abc', slot = 1)
```

Configures the slot offset k2 for the PUSCH, for the UL slot with the index <Slot>, for the initial BWP.

param cell_name
No help available

param slot
No help available

return
offset: No help available

set(*cell_name: str*, *slot: int*, *offset: int*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:UL:TDOMain:SOFFset
driver.configure.signaling.nradio.cell.ueScheduling.userDefined.sassignment.
↪uplink.tdomain.soffset.set(cell_name = 'abc', slot = 1, offset = 1)
```

Configures the slot offset k2 for the PUSCH, for the UL slot with the index <Slot>, for the initial BWP.

param cell_name
No help available

param slot
No help available

param offset
No help available

6.3.4.10.2.704 Symbol

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:UL:TDOMain:SYMBol
```

class SymbolCls

Symbol commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Number_Symbol: int: No parameter help available
- Start_Symbol: int: No parameter help available

get(*cell_name: str*, *slot: int*) → GetStruct

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:UL:TDOMain:SYMBol
value: GetStruct = driver.configure.signaling.nradio.cell.ueScheduling.
↪userDefined.sassignment.uplink.tdomain.symbol.get(cell_name = 'abc', slot = 1)
```

Defines the number of allocated OFDM symbols and the index of the first allocated OFDM symbol for the PDSCH. For the UL slot with the index <Slot>, for the initial BWP.

param cell_name
No help available

param slot
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *slot: int*, *number_symbol: int*, *start_symbol: int*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:UL:TDOMain:SYMBol
driver.configure.signaling.nradio.cell.ueScheduling.userDefined.sassignment.
↪uplink.tdomain.symbol.set(cell_name = 'abc', slot = 1, number_symbol = 1,␣
↪start_symbol = 1)
```

Defines the number of allocated OFDM symbols and the index of the first allocated OFDM symbol for the PDSCH. For the UL slot with the index <Slot>, for the initial BWP.

param cell_name
No help available

param slot
No help available

param number_symbol
No help available

param start_symbol
No help available

6.3.4.10.2.705 Tpmi

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:UL:TPMI
```

class TpmiCls

Tpmi commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *slot: int*) → int

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:UL:TPMI
value: int = driver.configure.signaling.nradio.cell.ueScheduling.userDefined.
↪sassignment.uplink.tpmi.get(cell_name = 'abc', slot = 1)
```

Specifies the TPMI for the UL slot with the index <Slot>, for the initial BWP.

param cell_name
No help available

param slot
No help available

return
tpmi: No help available

set(*cell_name: str*, *slot: int*, *tpmi: int*) → None

```
# SCPI:␣
↪[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:SASSignment:UL:TPMI
driver.configure.signaling.nradio.cell.ueScheduling.userDefined.sassignment.
↪uplink.tpmi.set(cell_name = 'abc', slot = 1, tpmi = 1)
```

Specifies the TPMI for the UL slot with the index <Slot>, for the initial BWP.

param cell_name
No help available

param slot
No help available

param tpmi
No help available

6.3.4.10.2.706 Uplink

class UplinkCls

Uplink commands group definition. 7 total commands, 7 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.ueScheduling.userDefined.uplink.clone()
```

Subgroups

6.3.4.10.2.707 Alevel

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:UL:ALEVel
```

class AlevelCls

Alevel commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → Level

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:UL:ALEVel
value: enums.Level = driver.configure.signaling.nradio.cell.ueScheduling.
↪userDefined.uplink.alevel.get(cell_name = 'abc')
```

Specifies the aggregation level for the UL, for the initial BWP.

param cell_name
No help available

return
level: No help available

set(*cell_name: str*, *level: Level*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:UL:ALEVel
driver.configure.signaling.nradio.cell.ueScheduling.userDefined.uplink.alevel.
↪set(cell_name = 'abc', level = enums.Level.AL1)
```

Specifies the aggregation level for the UL, for the initial BWP.

param cell_name
No help available

param level
No help available

6.3.4.10.2.708 Bpid

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:UL:BPID
```

class BpidCls

Bpid commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:UL:BPID
value: int = driver.configure.signaling.nradio.cell.ueScheduling.userDefined.
↪uplink.bpid.get(cell_name = 'abc')
```

Queries the ID of the UL bandwidth part that contains the scheduled, for the initial BWP.

param cell_name
No help available

return
idn: No help available

6.3.4.10.2.709 McsTable

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:UL:MCSTable
```

class McsTableCls

McsTable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → McsTableB

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:UL:MCSTable
value: enums.McsTableB = driver.configure.signaling.nradio.cell.ueScheduling.
↪userDefined.uplink.mcsTable.get(cell_name = 'abc')
```

Defines which MCS table must be used for PUSCH without transform precoding, for the initial BWP.

param cell_name
No help available

return
mcs_table: 256QAM, 64QAM low SE, 64QAM

set(*cell_name: str*, *mcs_table: McsTableB*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:UL:MCSTable
driver.configure.signaling.nradio.cell.ueScheduling.userDefined.uplink.mcsTable.
↪set(cell_name = 'abc', mcs_table = enums.McsTableB.L64)
```

Defines which MCS table must be used for PUSCH without transform precoding, for the initial BWP.

param cell_name
No help available

param mcs_table
256QAM, 64QAM low SE, 64QAM

6.3.4.10.2.710 PaFactor

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:UL:PAFactor
```

class PaFactorCls

PaFactor commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → UeScFactor

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:UL:PAFactor
value: enums.UeScFactor = driver.configure.signaling.nradio.cell.ueScheduling.
↪userDefined.uplink.paFactor.get(cell_name = 'abc')
```

Defines the 'pusch-AggregationFactor' of the PUSCH configuration, signaled to the UE, for the initial BWP.

param cell_name
No help available

return
factor: No help available

set(*cell_name: str*, *factor: UeScFactor*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:UL:PAFactor
driver.configure.signaling.nradio.cell.ueScheduling.userDefined.uplink.paFactor.
↪set(cell_name = 'abc', factor = enums.UeScFactor.N2)
```

Defines the 'pusch-AggregationFactor' of the PUSCH configuration, signaled to the UE, for the initial BWP.

param cell_name
No help available

param factor
No help available

6.3.4.10.2.711 PnoRepet

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:UL:PNORepet
```

class PnoRepetCls

PnoRepet commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → Repetitions

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:UL:PNORepet
value: enums.Repetitions = driver.configure.signaling.nradio.cell.ueScheduling.
↪userDefined.uplink.pnoRepet.get(cell_name = 'abc')
```

No command help available

param cell_name
No help available

return
repetitions: No help available

set(*cell_name: str*, *repetitions: Repetitions*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:UL:PNORepet
driver.configure.signaling.nradio.cell.ueScheduling.userDefined.uplink.pnoRepet.
↪set(cell_name = 'abc', repetitions = enums.Repetitions.N12)
```

No command help available

param cell_name
No help available

param repetitions
No help available

6.3.4.10.2.712 Prtype

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:UL:PRTYpe
```

class PrtypeCls

Prtype commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → Prtype

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:UL:PRTYpe
value: enums.Prtype = driver.configure.signaling.nradio.cell.ueScheduling.
↪userDefined.uplink.prtype.get(cell_name = 'abc')
```

Specifies the PUSCH repetition type signaled as ‘pusch-RepTypeIndicatorDCI-0-1’, for the initial BWP.

param cell_name
No help available

return
type_py: Not signaled, type A, type B.

set(*cell_name: str*, *type_py: Prtype*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:UL:PRTYpe
driver.configure.signaling.nradio.cell.ueScheduling.userDefined.uplink.prtype.
↪set(cell_name = 'abc', type_py = enums.Prtype.OFF)
```

Specifies the PUSCH repetition type signaled as ‘pusch-RepTypeIndicatorDCI-0-1’, for the initial BWP.

param cell_name
No help available

param type_py
Not signaled, type A, type B.

6.3.4.10.2.713 Ssid

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:UL:SSID
```

class SsidCls

Ssid commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:UL:SSID
value: int = driver.configure.signaling.nradio.cell.ueScheduling.userDefined.
↪uplink.ssid.get(cell_name = 'abc')
```

Configures the ID of the search space for the UL, for the initial BWP.

param cell_name
No help available

return
idn: No help available

set(*cell_name: str*, *idn: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UESCheduling:UDEFined:UL:SSID
driver.configure.signaling.nradio.cell.ueScheduling.userDefined.uplink.ssid.
↪set(cell_name = 'abc', idn = 1)
```

Configures the ID of the search space for the UL, for the initial BWP.

param cell_name
No help available

param idn
No help available

6.3.4.10.2.714 UeType

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UETYpe
```

class UeTypeCls

UeType commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → UeType

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UETYpe
value: enums.UeType = driver.configure.signaling.nradio.cell.ueType.get(cell_
↪name = 'abc')
```

Select the type of your UE (normal UE or RedCap UE) in edit mode.

param cell_name
No help available

return
ue_type: No help available

set(*cell_name: str*, *ue_type: UeType*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UETYpe
driver.configure.signaling.nradio.cell.ueType.set(cell_name = 'abc', ue_type =␣
↪enums.UeType.NORMal)
```

Select the type of your UE (normal UE or RedCap UE) in edit mode.

param cell_name
No help available

param ue_type
No help available

6.3.4.10.2.715 Uplink

class UplinkCls

Uplink commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.cell.uplink.clone()
```

Subgroups

6.3.4.10.2.716 LbWidth

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UL:LBWidth
```

class LbWidthCls

LbWidth commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UL:LBWidth
value: int = driver.configure.signaling.nradio.cell.uplink.lbWidth.get(cell_
↪name = 'abc')
```

Defines the resource indication value (RIV) signaled as 'locationAndBandwidth', for the uplink, for the initial BWP.

param cell_name
No help available

return
riv: No help available

set(*cell_name: str*, *riv: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UL:LBWidth
driver.configure.signaling.nradio.cell.uplink.lbWidth.set(cell_name = 'abc',␣
↪riv = 1)
```

Defines the resource indication value (RIV) signaled as 'locationAndBandwidth', for the uplink, for the initial BWP.

param cell_name
No help available

param riv
No help available

6.3.4.10.2.717 Mode

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UL:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → ConfigMode

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UL:MODE
value: enums.ConfigMode = driver.configure.signaling.nradio.cell.uplink.mode.
↪get(cell_name = 'abc')
```

Selects a configuration mode for the UL BWP settings in FDD, for the initial BWP.

param cell_name
No help available

return
mode: No help available

set(*cell_name: str*, *mode: ConfigMode*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UL:MODE
driver.configure.signaling.nradio.cell.uplink.mode.set(cell_name = 'abc', mode␣
↪= enums.ConfigMode.AUTO)
```

Selects a configuration mode for the UL BWP settings in FDD, for the initial BWP.

param cell_name
No help available

param mode
No help available

6.3.4.10.2.718 Rb

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:CELL:UL:RB
```

class RbCls

Rb commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Number_Rb: int: No parameter help available
- Start_Rb: int: No parameter help available

get(*cell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UL:RB
value: GetStruct = driver.configure.signaling.nradio.cell.uplink.rb.get(cell_
↪name = 'abc')
```

Defines the uplink of the initial BWP in the frequency domain as a contiguous set of RBs, within the carrier bandwidth.

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *number_rb: int*, *start_rb: int = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:CELL:UL:RB
driver.configure.signaling.nradio.cell.uplink.rb.set(cell_name = 'abc', number_
↪rb = 1, start_rb = 1)
```

Defines the uplink of the initial BWP in the frequency domain as a contiguous set of RBs, within the carrier bandwidth.

param cell_name
No help available

param number_rb
No help available

param start_rb
No help available

6.3.4.10.3 Ncell

class NcellCls

Ncell commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.nradio.ncell.clone()
```

Subgroups

6.3.4.10.3.1 Rcap

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:NCELl:RCAP
```

class RcapCls

Rcap commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *ncell_name: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:NRADio:NCELl:RCAP
value: bool = driver.configure.signaling.nradio.ncell.rcap.get(cell_name = 'abc
↪', ncell_name = 'abc')
```

Configures 'RedCapAccessallowed-r17' for an entry in the neighbor cell list of an NR cell.

param cell_name
Serving NR cell via which the neighbor cell list is broadcasted.

param ncell_name
Neighbor NR cell

return
enable: No help available

set(*cell_name: str*, *ncell_name: str*, *enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:NCELl:RCAP
driver.configure.signaling.nradio.ncell.rcap.set(cell_name = 'abc', ncell_name␣
↪= 'abc', enable = False)
```

Configures 'RedCapAccessallowed-r17' for an entry in the neighbor cell list of an NR cell.

param cell_name
Serving NR cell via which the neighbor cell list is broadcasted.

param ncell_name
Neighbor NR cell

param enable
No help available

6.3.4.10.3.2 Thresholds

SCPI Command :

```
[CONFigure]:SIGNaling:NRADio:NCELl:THResholds
```

class ThresholdsCls

Thresholds commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Threshold_Low: float: Threshold ThreshX Low ('ThreshX, LowP') .
- Threshold_High: float: Threshold ThreshX High ('ThreshX, HighP') .

get(*cell_name: str*, *ncell_name: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:NRADio:NCELl:THResholds
value: GetStruct = driver.configure.signaling.nradio.ncell.thresholds.get(cell_
↪name = 'abc', ncell_name = 'abc')
```

Configures reselection thresholds for an entry in the neighbor cell list of an LTE or NR cell.

param cell_name
Serving LTE or NR cell via which the neighbor cell list is broadcasted.

param ncell_name
Neighbor cell

return
structure: for return value, see the help for GetStruct structure arguments.

set(*cell_name: str*, *ncell_name: str*, *threshold_low: float*, *threshold_high: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:NRADio:NCELl:THResholds
driver.configure.signaling.nradio.ncell.thresholds.set(cell_name = 'abc', ncell_
↪name = 'abc', threshold_low = 1.0, threshold_high = 1.0)
```

Configures reselection thresholds for an entry in the neighbor cell list of an LTE or NR cell.

param cell_name
Serving LTE or NR cell via which the neighbor cell list is broadcasted.

param ncell_name
Neighbor cell

param threshold_low
Threshold ThreshX Low ('ThreshX, LowP') .

param threshold_high
Threshold ThreshX High ('ThreshX, HighP') .

6.3.4.11 Sms

SCPI Command :

```
[CONFigure]:SIGNaling:SMS:SCENtre
```

class SmsCls

Sms commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_scentre() → str

```
# SCPI: [CONFigure]:SIGNaling:SMS:SCENtre
value: str = driver.configure.signaling.sms.get_scentre()
```

Configures the number of the short message service center used by the originator of the message.

return
address: No help available

set_scentre(*address: str*) → None

```
# SCPI: [CONFigure]:SIGNaling:SMS:SCENtre
driver.configure.signaling.sms.set_scentre(address = 'abc')
```

Configures the number of the short message service center used by the originator of the message.

param address
No help available

6.3.4.12 Tmode

SCPI Commands :

```
[CONFigure]:SIGNaling:TMODe:TLOop
[CONFigure]:SIGNaling:TMODe
```

class TmodeCls

Tmode commands group definition. 5 total commands, 3 Subgroups, 2 group commands

get_tloop() → TestLoopState

```
# SCPI: [CONFigure]:SIGNaling:TMODe:TLOop
value: enums.TestLoopState = driver.configure.signaling.tmode.get_tloop()
```

Opens or closes a test loop mode A at the UE. Prerequisites: Test mode active and UE registered.

return
test_loop_state: No help available

get_value() → bool

```
# SCPI: [CONFigure]:SIGNaling:TMODe
value: bool = driver.configure.signaling.tmode.get_value()
```

No command help available

return
enable: No help available

set_tloop(*test_loop_state: TestLoopState*) → None

```
# SCPI: [CONFigure]:SIGNaling:TMODe:TLOop
driver.configure.signaling.tmode.set_tloop(test_loop_state = enums.
↪TestLoopState.CLOSe)
```

Opens or closes a test loop mode A at the UE. Prerequisites: Test mode active and UE registered.

param test_loop_state
No help available

set_value(*enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:TMODe
driver.configure.signaling.tmode.set_value(enable = False)
```

No command help available

param enable
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.tmode.clone()
```

Subgroups

6.3.4.12.1 Block

SCPI Command :

```
[CONFigure]:SIGNaling:TMODe:BLOCk
```

class BlockCls

Block commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class BlockStruct

Response structure. Fields:

- Enable: bool: Enable / disable the beamlock function.
- Test_Function: enums.TestFunction: RX: Beamlock for UE receiver beams. TX: Beamlock for UE transmitter beams. RXTX: Beamlock for UE receiver and transmitter beams.

get() → BlockStruct

```
# SCPI: [CONFigure]:SIGNaling:TMODe:BLOCk
value: BlockStruct = driver.configure.signaling.tmode.block.get()
```

Enables or disables the beamlock function of the UE and selects the direction for enabled beamlock.

return
structure: for return value, see the help for BlockStruct structure arguments.

set(*enable: bool*, *test_function: TestFunction = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:TMODe:BLOCk
driver.configure.signaling.tmode.block.set(enable = False, test_function =␣
↪enums.TestFunction.RX)
```

Enables or disables the beamlock function of the UE and selects the direction for enabled beamlock.

param enable
Enable / disable the beamlock function.

param test_function
RX: Beamlock for UE receiver beams. TX: Beamlock for UE transmitter beams. RXTX: Beamlock for UE receiver and transmitter beams.

6.3.4.12.2 SsReport

SCPI Command :

```
[CONFigure]:SIGNaling:TMODe:SSReport:ENABle
```

class SsReportCls

SsReport commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_enable() → bool

```
# SCPI: [CONFigure]:SIGNaling:TMODe:SSReport:ENABle
value: bool = driver.configure.signaling.tmode.ssReport.get_enable()
```

Enables SS-RSRPB reporting by the UE.

return
enable: No help available

set_enable(*enable: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:TMODe:SSReport:ENABle
driver.configure.signaling.tmode.ssReport.set_enable(enable = False)
```

Enables SS-RSRPB reporting by the UE.

param enable
No help available

6.3.4.12.3 UepLimit

SCPI Command :

```
[CONFigure]:SIGNaling:TMODe:UEPLimit
```

class UepLimitCls

UepLimit commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Status: enums.LimitStatus: OFF: UPLF is off. DPRogress: Deactivation of UPLF is in progress. ON: UPLF is ON. APRogress: Activation of UPLF is in progress.
- Pcell_Nr: enums.PcellNr: Value in MHz, configuring the information element 'PCELL NR BANDWIDTH' of the 'ACTIVATE POWER LIMIT REQUEST' message.
- Bwidth_Total: enums.BwidthTotal: Value in MHz, configuring the information element 'TOTAL NR AGGREGATED BANDWIDTH' of the 'ACTIVATE POWER LIMIT REQUEST' message.

get() → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:TMODe:UEPLimit
value: GetStruct = driver.configure.signaling.tmode.uepLimit.get()
```

Enables or disables the UL power limit function (UPLF) test mode at the UE.

return
structure: for return value, see the help for GetStruct structure arguments.

set(*enable: bool*, *pcell_nr: PcellNr = None*, *bwidth_total: BwidthTotal = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:TMODe:UEPLimit
driver.configure.signaling.tmode.uepLimit.set(enable = False, pcell_nr = enums.
↪PcellNr.B050, bwidth_total = enums.BwidthTotal.B100)
```

Enables or disables the UL power limit function (UPLF) test mode at the UE.

param enable

- ON: Send an 'ACTIVATE POWER LIMIT REQUEST' message.
- OFF: Send a 'DEACTIVATE POWER LIMIT REQUEST' message.

param pcell_nr
Value in MHz, configuring the information element 'PCELL NR BANDWIDTH' of the 'ACTIVATE POWER LIMIT

REQUEST' message. :param bwidth_total: Value in MHz, configuring the information element 'TOTAL NR AGGREGATED BANDWIDTH' of the 'ACTIVATE POWER LIMIT REQUEST' message.

6.3.4.13 Topology

class TopologyCls

Topology commands group definition. 16 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.topology.clone()
```

Subgroups

6.3.4.13.1 Eps

class EpsCls

Eps commands group definition. 5 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.topology.eps.clone()
```

Subgroups

6.3.4.13.1.1 Info

SCPI Command :

```
[CONFigure]:SIGNaling:TOPology:EPS:INFO
```

class InfoCls

Info commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Ta_Code: int: Tracking area code (TAC) .
- Timer_3412: float: No effect - for future use.
- Count_Cells_Lte: int: Number of associated LTE cells.
- Count_Cells_Nr: int: Number of associated NR cells.
- List_Of_Cells_Lte: int: Comma-separated list of strings, one string per LTE cell (name of the cell) . If there are no LTE cells, an empty string is returned.
- List_Of_Cells_Nr: int: Comma-separated list of strings, one string per NR cell (name of the cell) . If there are no NR cells, an empty string is returned.

get(*name_ta_eps: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:TOPology:EPS:INFO
value: GetStruct = driver.configure.signaling.topology.eps.info.get(name_ta_eps
↪= 'abc')
```

Queries basic information about an EPS tracking area.

param name_ta_eps
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

6.3.4.13.1.2 TaCode

SCPI Command :

```
[CONFigure]:SIGNaling:TOPology:EPS:TACode
```

class TaCodeCls

TaCode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*name_ta_eps: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:TOPology:EPS:TACode
value: int = driver.configure.signaling.topology.eps.taCode.get(name_ta_eps =
↪'abc')
```

Configures the tracking area code (TAC) of an EPS tracking area.

param name_ta_eps
No help available

return
ta_code: No help available

set(*name_ta_eps: str*, *ta_code: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:TOPology:EPS:TACode
driver.configure.signaling.topology.eps.taCode.set(name_ta_eps = 'abc', ta_code
↪= 1)
```

Configures the tracking area code (TAC) of an EPS tracking area.

param name_ta_eps
No help available

param ta_code
No help available

6.3.4.13.1.3 Timer

SCPI Command :

```
[CONFigure]:SIGNaling:TOPology:EPS:TIMer
```

class TimerCls

Timer commands group definition. 3 total commands, 1 Subgroups, 1 group commands

get(*name_ta_eps: str*) → float

```
# SCPI: [CONFigure]:SIGNaling:TOPology:EPS:TIMer
value: float = driver.configure.signaling.topology.eps.timer.get(name_ta_eps =
↪'abc')
```

No command help available

param name_ta_eps
No help available

return
timer: No help available

set(*name_ta_eps: str*, *timer: float*) → None

```
# SCPI: [CONFigure]:SIGNaling:TOPology:EPS:TIMer
driver.configure.signaling.topology.eps.timer.set(name_ta_eps = 'abc', timer =␣
↪1.0)
```

No command help available

param name_ta_eps
No help available

param timer
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.topology.eps.timer.clone()
```

Subgroups

6.3.4.13.1.4 T<Tnum>

RepCap Settings

```
# Range: Nr300 .. Nr319
rc = driver.configure.signaling.topology.eps.timer.t.repcap_tnum_get()
driver.configure.signaling.topology.eps.timer.t.repcap_tnum_set(repcap.Tnum.Nr300)
```

SCPI Command :

```
[CONFigure]:SIGNaling:TOPology:EPS:TIMer:T<no>
```

class TCls

T commands group definition. 2 total commands, 1 Subgroups, 1 group commands Repeated Capability: Tnum, default value after init: Tnum.Nr300

class GetStruct

Response structure. Fields:

- Factor: int: The timer value is calculated as Factor * Unit.
- Unit: enums.TimerUnitB: S2: unit 2 seconds M1: unit 1 minute M6: unit 6 minutes DEACtivated: timer deactivated

get(*name_ta_eps: str*, *tnum=Tnum.Default*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:TOPology:EPS:TIMer:T<no>
value: GetStruct = driver.configure.signaling.topology.eps.timer.t.get(name_ta_
↪eps = 'abc', tnum = repcap.Tnum.Default)
```

Configures the timer T3412 (periodic EPS tracking area update) .

param name_ta_eps
Name of EPS tracking area

param tnum
optional repeated capability selector. Default value: Nr300 (settable in the interface 'T')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*name_ta_eps: str*, *factor: int*, *unit: TimerUnitB = None*, *tnum=Tnum.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:TOPology:EPS:TIMer:T<no>
driver.configure.signaling.topology.eps.timer.t.set(name_ta_eps = 'abc', factor␣
↪= 1, unit = enums.TimerUnitB.DEACtivated, tnum = repcap.Tnum.Default)
```

Configures the timer T3412 (periodic EPS tracking area update) .

param name_ta_eps
Name of EPS tracking area

param factor
The timer value is calculated as Factor * Unit.

param unit
S2: unit 2 seconds M1: unit 1 minute M6: unit 6 minutes DEACtivated: timer deactivated

param tnum
optional repeated capability selector. Default value: Nr300 (settable in the interface 'T')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.topology.eps.timer.t.clone()
```

Subgroups

6.3.4.13.1.5 Extended

SCPI Command :

```
[CONFigure]:SIGNaling:TOPology:EPS:TIMer:T<no>:EXTended
```

class ExtendedCls

Extended commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Enable: bool: ON: Send the timer value to the UE. OFF: Do not send a timer value.
- Factor: int: The timer value is calculated as Factor * Unit.
- Unit: enums.TimerUnit: S2, S30: unit 2 seconds, 30 seconds M1, M10: unit 1 minute, 10 minutes H1, H10, H320: unit 1 hour, 10 hours, 320 hours DEACtivated: timer deactivated

get(*name_ta_eps: str*, *tnum=Tnum.Default*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:TOPology:EPS:TIMer:T<no>:EXTended
value: GetStruct = driver.configure.signaling.topology.eps.timer.t.extended.
↪get(name_ta_eps = 'abc', tnum = repcap.Tnum.Default)
```

Configures the extended timer T3412.

param name_ta_eps
Name of EPS tracking area

param tnum
optional repeated capability selector. Default value: Nr300 (settable in the interface 'T')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*name_ta_eps: str*, *enable: bool*, *factor: int = None*, *unit: TimerUnit = None*, *tnum=Tnum.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:TOPology:EPS:TIMer:T<no>:EXTended
driver.configure.signaling.topology.eps.timer.t.extended.set(name_ta_eps = 'abc
↪', enable = False, factor = 1, unit = enums.TimerUnit.DEACtivated, tnum =␣
↪repcap.Tnum.Default)
```

Configures the extended timer T3412.

param name_ta_eps
Name of EPS tracking area

param enable
ON: Send the timer value to the UE. OFF: Do not send a timer value.

param factor
The timer value is calculated as Factor * Unit.

param unit
S2, S30: unit 2 seconds, 30 seconds M1, M10: unit 1 minute, 10 minutes H1, H10, H320: unit 1 hour, 10 hours, 320 hours DEACtivated: timer deactivated

param tnum
optional repeated capability selector. Default value: Nr300 (settable in the interface 'T')

6.3.4.13.2 Fgs

class FgsCls

Fgs commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.topology.fgs.clone()
```

Subgroups

6.3.4.13.2.1 Default

SCPI Command :

```
[CONFigure]:SIGNaling:TOPology:FGS:DEFault:VOICe
```

class DefaultCls

Default commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_voice() → VoiceHandling

```
# SCPI: [CONFigure]:SIGNaling:TOPology:FGS:DEFault:VOICe
value: enums.VoiceHandling = driver.configure.signaling.topology.fgs.default.
↪get_voice()
```

Defines the handling of voice calls for UE registered in a 5GS tracking area.

return
voice_handling: UECap: The fallback decision is based on UE capabilities. VONR: Always voice over NR. EFRedirect: Always EPS fallback with redirection. EFHandover: Always EPS fallback with handover.

set_voice(*voice_handling: VoiceHandling*) → None

```
# SCPI: [CONFigure]:SIGNaling:TOPology:FGS:DEFault:VOICe
driver.configure.signaling.topology.fgs.default.set_voice(voice_handling =␣
↪enums.VoiceHandling.EFHandover)
```

Defines the handling of voice calls for UE registered in a 5GS tracking area.

param voice_handling
UECap: The fallback decision is based on UE capabilities. VONR: Always voice over NR. EFRedirect: Always EPS fallback with redirection. EFHandover: Always EPS fallback with handover.

6.3.4.13.2.2 Info

SCPI Command :

```
[CONFigure]:SIGNaling:TOPology:FGS:INFO
```

class InfoCls

Info commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Ta_Code: str: Tracking area code (TAC) .
- Count_Cells_Lte: int: Number of associated LTE cells.
- Count_Cells_Nr: int: Number of associated NR cells.
- List_Of_Cells_Lte: str: Comma-separated list of strings, one string per LTE cell (name of the cell) . If there are no LTE cells, an empty string is returned.
- List_Of_Cells_Nr: str: Comma-separated list of strings, one string per NR cell (name of the cell) . If there are no NR cells, an empty string is returned.

get(*name_ta_5_g: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:TOPology:FGS:INFO
value: GetStruct = driver.configure.signaling.topology.fgs.info.get(name_ta_5_g␣
↪= 'abc')
```

Queries basic information about a 5GS tracking area.

param name_ta_5_g
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

6.3.4.13.2.3 TaCode

SCPI Command :

```
[CONFigure]:SIGNaling:TOPology:FGS:TACode
```

class TaCodeCls

TaCode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*name_ta_5_g: str*) → int

```
# SCPI: [CONFigure]:SIGNaling:TOPology:FGS:TACode
value: int = driver.configure.signaling.topology.fgs.taCode.get(name_ta_5_g =
↪'abc')
```

Configures the tracking area code (TAC) of a 5GS tracking area.

param name_ta_5_g
No help available

return
ta_code: No help available

set(*name_ta_5_g: str*, *ta_code: int*) → None

```
# SCPI: [CONFigure]:SIGNaling:TOPology:FGS:TACode
driver.configure.signaling.topology.fgs.taCode.set(name_ta_5_g = 'abc', ta_code␣
↪= 1)
```

Configures the tracking area code (TAC) of a 5GS tracking area.

param name_ta_5_g
No help available

param ta_code
No help available

6.3.4.13.2.4 Timer

class TimerCls

Timer commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.topology.fgs.timer.clone()
```

Subgroups

6.3.4.13.2.5 T<Tnum>

RepCap Settings

```
# Range: Nr300 .. Nr319
rc = driver.configure.signaling.topology.fgs.timer.t.repcap_tnum_get()
driver.configure.signaling.topology.fgs.timer.t.repcap_tnum_set(repcap.Tnum.Nr300)
```

SCPI Command :

```
[CONFigure]:SIGNaling:TOPology:FGS:TIMer:T<no>
```

class TCls

T commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Tnum, default value after init: Tnum.Nr300

class GetStruct

Response structure. Fields:

- Factor: int: The timer value is calculated as Factor * Unit.
- Unit: enums.TimerUnit: S2, S30: unit 2 seconds, 30 seconds M1, M10: unit 1 minute, 10 minutes H1, H10, H320: unit 1 hour, 10 hours, 320 hours DEACtivated: timer deactivated

get(*name_ta_5_g: str*, *tnum=Tnum.Default*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:TOPology:FGS:TIMer:T<no>
value: GetStruct = driver.configure.signaling.topology.fgs.timer.t.get(name_ta_
↪5_g = 'abc', tnum = repcap.Tnum.Default)
```

Configures the timer T3512 (periodic registration update in a 5GS tracking area) .

param name_ta_5_g
Name of 5GS tracking area

param tnum
optional repeated capability selector. Default value: Nr300 (settable in the interface 'T')

return
structure: for return value, see the help for GetStruct structure arguments.

set(*name_ta_5_g: str*, *factor: int*, *unit: TimerUnit = None*, *tnum=Tnum.Default*) → None

```
# SCPI: [CONFigure]:SIGNaling:TOPology:FGS:TIMer:T<no>
driver.configure.signaling.topology.fgs.timer.t.set(name_ta_5_g = 'abc', factor␣
↪= 1, unit = enums.TimerUnit.DEACtivated, tnum = repcap.Tnum.Default)
```

Configures the timer T3512 (periodic registration update in a 5GS tracking area) .

param name_ta_5_g
Name of 5GS tracking area

param factor
The timer value is calculated as Factor * Unit.

param unit
S2, S30: unit 2 seconds, 30 seconds M1, M10: unit 1 minute, 10 minutes H1, H10, H320: unit 1 hour, 10 hours, 320 hours DEACtivated: timer deactivated

param tnum
optional repeated capability selector. Default value: Nr300 (settable in the interface 'T')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.topology.fgs.timer.t.clone()
```

6.3.4.13.2.6 Ue

class UeCls

Ue commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.topology.fgs.ue.clone()
```

Subgroups

6.3.4.13.2.7 Pdu

class PduCls

Pdu commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.topology.fgs.ue.pdu.clone()
```

Subgroups

6.3.4.13.2.8 QosFlow

SCPI Command :

```
[CONFigure]:SIGNaling:TOPology:FGS:UE:PDU:QOSFlow
```

class QosFlowCls

QosFlow commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Qi: enums.Qi: 5G quality of service identifier (5QI) .
- Max_Dl_Bitrate: int: Maximum flow bit rate (MFBR) for the DL.
- Max_Dl_Unit: enums.ItRateUnit: Unit for MaxDLBitrate. Kn, Mn, Gn, Tn, Pn = n kbit/s, Mbit/s, Gbit/s, Tbit/s, Pbit/s

- Max_Ul_Bitrate: int: Maximum flow bit rate (MFBR) for the UL.
- Max_Ul_Unit: enums.ItRateUnit: Unit for MaxULBitrate.
- Flow_Control: enums.FlowControl: GUARanteed: GBR QoS flow NGUaranteed: non-GBR QoS flow
- Dl_Bitrate: int: Guaranteed flow bit rate (GFBR) for the DL, only for GBR QoS flows.
- Dl_Unit: enums.ItRateUnit: Unit for DLBitrate, only for GBR QoS flows.
- Ul_Bitrate: int: Guaranteed flow bit rate (GFBR) for the UL, only for GBR QoS flows.
- Ul_Unit: enums.ItRateUnit: Unit for ULBitrate, only for GBR QoS flows.
- Averaging_Window: int or bool: Duration over which the bit rates GFBR and MFBR are calculated for GBR QoS flows. OFF omits the parameter in the QoS flow description.

class SetStruct

Structure for setting input parameters. Contains optional setting parameters. Fields:

- Ue_Id: str: For future use. Enter any value.
- Qos_Flow_Id: int: ID of the QoS flow to be modified.
- Qi: enums.Qi: Optional setting parameter. 5G quality of service identifier (5QI) .
- Max_Dl_Bitrate: int: Optional setting parameter. Maximum flow bit rate (MFBR) for the DL.
- Max_Dl_Unit: enums.ItRateUnit: Optional setting parameter. Unit for MaxDLBitrate. Kn, Mn, Gn, Tn, Pn = n kbit/s, Mbit/s, Gbit/s, Tbit/s, Pbit/s
- Max_Ul_Bitrate: int: Optional setting parameter. Maximum flow bit rate (MFBR) for the UL.
- Max_Ul_Unit: enums.ItRateUnit: Optional setting parameter. Unit for MaxULBitrate.
- Flow_Control: enums.FlowControl: Optional setting parameter. GUARanteed: GBR QoS flow NGUaranteed: non-GBR QoS flow
- Dl_Bitrate: int: Optional setting parameter. Guaranteed flow bit rate (GFBR) for the DL, only for GBR QoS flows.
- Dl_Unit: enums.ItRateUnit: Optional setting parameter. Unit for DLBitrate, only for GBR QoS flows.
- Ul_Bitrate: int: Optional setting parameter. Guaranteed flow bit rate (GFBR) for the UL, only for GBR QoS flows.
- Ul_Unit: enums.ItRateUnit: Optional setting parameter. Unit for ULBitrate, only for GBR QoS flows.
- Averaging_Window: int or bool: Optional setting parameter. Duration over which the bit rates GFBR and MFBR are calculated for GBR QoS flows. OFF omits the parameter in the QoS flow description.

get(*ue_id: str*, *qos_flow_id: int*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:TOPology:FGS:UE:PDU:QOSFlow
value: GetStruct = driver.configure.signaling.topology.fgs.ue.pdu.qosFlow.
↪get(ue_id = 'abc', qos_flow_id = 1)
```

Modifies an existing QoS flow.

param ue_id
For future use. Enter any value.

param qos_flow_id
ID of the QoS flow to be modified.

return
structure: for return value, see the help for GetStruct structure arguments.

set(*structure: SetStruct*) → None

```
# SCPI: [CONFigure]:SIGNaling:TOPology:FGS:UE:PDU:QOSFlow
structure = driver.configure.signaling.topology.fgs.ue.pdu.qosFlow.SetStruct()
structure.Ue_Id: str = 'abc'
structure.Qos_Flow_Id: int = 1
structure.Qi: enums.Qi = enums.Qi.Q1
structure.Max_Dl_Bitrate: int = 1
structure.Max_Dl_Unit: enums.ItRateUnit = enums.ItRateUnit.G1
structure.Max_Ul_Bitrate: int = 1
structure.Max_Ul_Unit: enums.ItRateUnit = enums.ItRateUnit.G1
structure.Flow_Control: enums.FlowControl = enums.FlowControl.GUARanteed
structure.Dl_Bitrate: int = 1
structure.Dl_Unit: enums.ItRateUnit = enums.ItRateUnit.G1
structure.Ul_Bitrate: int = 1
structure.Ul_Unit: enums.ItRateUnit = enums.ItRateUnit.G1
structure.Averaging_Window: int or bool = 1
driver.configure.signaling.topology.fgs.ue.pdu.qosFlow.set(structure)
```

Modifies an existing QoS flow.

param structure
for set value, see the help for SetStruct structure arguments.

6.3.4.13.3 Plmn

class PlmnCls

Plmn commands group definition. 6 total commands, 6 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.topology.plmn.clone()
```

Subgroups

6.3.4.13.3.1 EpsFallback

SCPI Command :

```
[CONFigure]:SIGNaling:TOPology:PLMN:EPSFallback
```

class EpsFallbackCls

EpsFallback commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*name_plmn: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:TOPology:PLMN:EPSFallback
value: bool = driver.configure.signaling.topology.plmn.epsFallback.get(name_
↪plmn = 'abc')
```

Selects whether the 4G MME behind LTE cells has an N26 interface to a 5G AMF.

param name_plmn
No help available

return
n_26_support: No help available

set(*name_plmn: str*, *n_26_support: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:TOPology:PLMN:EPSFallback
driver.configure.signaling.topology.plmn.epsFallback.set(name_plmn = 'abc', n_
↪26_support = False)
```

Selects whether the 4G MME behind LTE cells has an N26 interface to a 5G AMF.

param name_plmn
No help available

param n_26_support
No help available

6.3.4.13.3.2 FgsFallback

SCPI Command :

```
[CONFigure]:SIGNaling:TOPology:PLMN:FGSFallback
```

class FgsFallbackCls

FgsFallback commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*name_plmn: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:TOPology:PLMN:FGSFallback
value: bool = driver.configure.signaling.topology.plmn.fgsFallback.get(name_
↪plmn = 'abc')
```

Selects whether the 5G AMF behind NR cells has an N26 interface to a 4G MME.

param name_plmn
No help available

return
n_26_support: No help available

set(*name_plmn: str*, *n_26_support: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:TOPology:PLMN:FGSFallback
driver.configure.signaling.topology.plmn.fgsFallback.set(name_plmn = 'abc', n_
↪26_support = False)
```

Selects whether the 5G AMF behind NR cells has an N26 interface to a 4G MME.

param name_plmn
No help available

param n_26_support
No help available

6.3.4.13.3.3 Info

SCPI Command :

```
[CONFigure]:SIGNaling:TOPology:PLMN:INFO
```

class InfoCls

Info commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Mcc: str: No parameter help available
- Mnc: str: No parameter help available
- Count_Ta_Eps: int: Number of associated EPS tracking areas.
- Count_Ta_5_G: int: Number of associated 5GS tracking areas.
- List_Name_Ta_Eps: str: Comma-separated list of strings, one string per EPS tracking area (name of the EPS TA) . If there are no EPS TAs, an empty string is returned.
- List_Name_5_G: str: Comma-separated list of strings, one string per 5GS tracking area (name of the 5GS TA) . If there are no 5GS TAs, an empty string is returned.

get(*name_plmn: str*) → GetStruct

```
# SCPI: [CONFigure]:SIGNaling:TOPology:PLMN:INFO
value: GetStruct = driver.configure.signaling.topology.plmn.info.get(name_plmn
↪= 'abc')
```

Queries basic information about a PLMN.

param name_plmn
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

6.3.4.13.3.4 Mcc

SCPI Command :

```
[CONFigure]:SIGNaling:TOPology:PLMN:MCC
```

class MccCls

Mcc commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*name_plmn: str*) → str

```
# SCPI: [CONFigure]:SIGNaling:TOPology:PLMN:MCC
value: str = driver.configure.signaling.topology.plmn.mcc.get(name_plmn = 'abc')
```

Configures the mobile country code (MCC) of a PLMN.

param name_plmn
No help available

return
mcc: No help available

set(*name_plmn: str*, *mcc: str*) → None

```
# SCPI: [CONFigure]:SIGNaling:TOPology:PLMN:MCC
driver.configure.signaling.topology.plmn.mcc.set(name_plmn = 'abc', mcc = 'abc')
```

Configures the mobile country code (MCC) of a PLMN.

param name_plmn
No help available

param mcc
No help available

6.3.4.13.3.5 Mnc

SCPI Command :

```
[CONFigure]:SIGNaling:TOPology:PLMN:MNC
```

class MncCls

Mnc commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*name_plmn: str*) → str

```
# SCPI: [CONFigure]:SIGNaling:TOPology:PLMN:MNC
value: str = driver.configure.signaling.topology.plmn.mnc.get(name_plmn = 'abc')
```

Configures the mobile network code (MNC) of a PLMN.

param name_plmn
No help available

return
mnc: No help available

set(*name_plmn: str*, *mnc: str*) → None

```
# SCPI: [CONFigure]:SIGNaling:TOPology:PLMN:MNC
driver.configure.signaling.topology.plmn.mnc.set(name_plmn = 'abc', mnc = 'abc')
```

Configures the mobile network code (MNC) of a PLMN.

param name_plmn
No help available

param mnc
No help available

6.3.4.13.3.6 SmeBearers

SCPI Command :

```
[CONFigure]:SIGNaling:TOPology:PLMN:SMEBearers
```

class SmeBearersCls

SmeBearers commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*name_plmn: str*) → bool

```
# SCPI: [CONFigure]:SIGNaling:TOPology:PLMN:SMEBearers
value: bool = driver.configure.signaling.topology.plmn.smeBearers.get(name_plmn␣
↪= 'abc')
```

Selects whether the network supports up to 15 EPS bearer contexts per UE (or only up to 8 EPS bearer contexts) .

param name_plmn
No help available

return
max_15_support: No help available

set(*name_plmn: str*, *max_15_support: bool*) → None

```
# SCPI: [CONFigure]:SIGNaling:TOPology:PLMN:SMEBearers
driver.configure.signaling.topology.plmn.smeBearers.set(name_plmn = 'abc', max_
↪15_support = False)
```

Selects whether the network supports up to 15 EPS bearer contexts per UE (or only up to 8 EPS bearer contexts) .

param name_plmn
No help available

param max_15_support
No help available

6.3.4.14 Trigger

SCPI Command :

```
CONFigure:SIGNaling:TRIGger:SCOPe
```

class TriggerCls

Trigger commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_scope() → List[str]

```
# SCPI: CONFigure:SIGNaling:TRIGger:SCOPe
value: List[str] = driver.configure.signaling.trigger.get_scope()
```

Activates one or more trigger types. You can query all inactive trigger types via method RsCMX_Signaling.Catalog. Signaling.Trigger.source.

return
trigger: Comma-separated list of strings, one string per trigger type to be activated.

set_scope(*trigger: List[str]*) → None

```
# SCPI: CONFigure:SIGNaling:TRIGger:SCOPe
driver.configure.signaling.trigger.set_scope(trigger = ['abc1', 'abc2', 'abc3'])
```

Activates one or more trigger types. You can query all inactive trigger types via method RsCMX_Signaling.Catalog. Signaling.Trigger.source.

param trigger
Comma-separated list of strings, one string per trigger type to be activated.

6.3.4.15 Ue

class UeCls

Ue commands group definition. 5 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.ue.clone()
```

Subgroups

6.3.4.15.1 Rrc

class RrcCls

Rrc commands group definition. 5 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.ue.rrc.clone()
```

Subgroups

6.3.4.15.1.1 Asn

SCPI Commands :

```
[CONFigure]:SIGNaling:UE:RRC:ASN:SETup
[CONFigure]:SIGNaling:UE:RRC:ASN:RECOnfig
[CONFigure]:SIGNaling:UE:RRC:ASN:RELease
```

class AsnCls

Asn commands group definition. 5 total commands, 1 Subgroups, 3 group commands

set_re_config(*message: str*) → None

```
# SCPI: [CONFigure]:SIGNaling:UE:RRC:ASN:RECOnfig
driver.configure.signaling.ue.rrc.asn.set_re_config(message = 'abc')
```

No command help available

param message
No help available

set_release(*message: str*) → None

```
# SCPI: [CONFigure]:SIGNaling:UE:RRC:ASN:RELease
driver.configure.signaling.ue.rrc.asn.set_release(message = 'abc')
```

No command help available

param message
No help available

set_setup(*message: str*) → None

```
# SCPI: [CONFigure]:SIGNaling:UE:RRC:ASN:SETup
driver.configure.signaling.ue.rrc.asn.set_setup(message = 'abc')
```

No command help available

param message
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.ue.rrc.asn.clone()
```

Subgroups

6.3.4.15.1.2 Lte

SCPI Commands :

```
[CONFigure]:SIGNaling:UE:RRC:ASN:LTE:REConfig
[CONFigure]:SIGNaling:UE:RRC:ASN:LTE:RELease
```

class LteCls

Lte commands group definition. 2 total commands, 0 Subgroups, 2 group commands

set_re_config(*message: str*) → None

```
# SCPI: [CONFigure]:SIGNaling:UE:RRC:ASN:LTE:REConfig
driver.configure.signaling.ue.rrc.asn.lte.set_re_config(message = 'abc')
```

No command help available

param message
No help available

set_release(*message: str*) → None

```
# SCPI: [CONFigure]:SIGNaling:UE:RRC:ASN:LTE:RELease
driver.configure.signaling.ue.rrc.asn.lte.set_release(message = 'abc')
```

No command help available

param message
No help available

6.3.4.16 UeAssistance

class UeAssistanceCls

UeAssistance commands group definition. 9 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.ueAssistance.clone()
```

Subgroups

6.3.4.16.1 Nradio

SCPI Command :

```
[CONFigure]:SIGNaling:UEASsistance:NRADio
```

class NradioCls

Nradio commands group definition. 9 total commands, 8 Subgroups, 1 group commands

class ValueStruct

Structure for setting input parameters. Contains optional set arguments. Fields:

- Dbr_Enable: bool: Enables/disables transmitting the parameter 'delayBudgetReportingConfig' in the IE 'OtherConfig'.
- Dbr_Prohibit_Timer: enums.ProhibitTimer: Optional setting parameter. Signaled 'delayBudgetReportingProhibitTimer'. Sn: n ms SnDm: n.m ms
- Oass_Enable: bool: Optional setting parameter. Enables/disables transmitting the parameter 'OverheatingAssistanceConfig' in the IE 'OtherConfig'.
- Oass_Prohibit_Timer: enums.ProhibitTimer: Optional setting parameter. Signaled 'overheatingIndicationProhibitTimer'. Sn: n ms SnDm: n.m ms
- Drxp_Enable: bool: Optional setting parameter. Enables/disables transmitting the parameter 'DRX-PreferenceConfig-r16' in the IE 'OtherConfig'.
- Drxp_Prohibit_Timer: enums.ProhibitTimer: Optional setting parameter. Signaled 'drx-PreferenceProhibitTimer-r16'. Sn: n ms SnDm: n.m ms
- Mbwp_Enable: bool: Optional setting parameter. Enables/disables transmitting the parameter 'MaxBW-PreferenceConfig-r16' in the IE 'OtherConfig'.
- Mbwp_Prohibit_Timer: enums.ProhibitTimer: Optional setting parameter. Signaled 'maxBW-PreferenceProhibitTimer-r16'. Sn: n ms SnDm: n.m ms
- Mccp_Enable: bool: Optional setting parameter. Enables/disables transmitting the parameter 'MaxCC-PreferenceConfig-r16' in the IE 'OtherConfig'.
- Mccp_Prohibit_Timer: enums.ProhibitTimer: Optional setting parameter. Signaled 'maxCC-PreferenceProhibitTimer-r16'. Sn: n ms SnDm: n.m ms
- Mml_Enable: bool: Optional setting parameter. Enables/disables transmitting the parameter 'MaxMIMO-LayerPreferenceConfig-r16' in the IE 'OtherConfig'.
- Mml_Prohibit_Timer: enums.ProhibitTimer: Optional setting parameter. Signaled 'maxMIMO-LayerPreferenceProhibitTimer-r16'. Sn: n ms SnDm: n.m ms
- Msof_Enable: bool: Optional setting parameter. Enables/disables transmitting the parameter 'MinSchedulingOffsetPreferenceConfig-r16' in the IE 'OtherConfig'.
- Msof_Prohibit_Timer: enums.ProhibitTimer: Optional setting parameter. Signaled 'minSchedulingOffsetPreferenceProhibitTimer-r16'. Sn: n ms SnDm: n.m ms
- Relp_Enable: bool: Optional setting parameter. Enables/disables transmitting the parameter 'ReleasePreferenceConfig-r16' in the IE 'OtherConfig'.
- Relp_Prohibit_Timer: enums.ProhibitTimer: Optional setting parameter. Signaled 'releasePreferenceProhibitTimer-r16'. Sn: n ms SnDm: n.m ms

get_value() → ValueStruct

```
# SCPI: [CONFigure]:SIGNaling:UEASsistance:NRADio
value: ValueStruct = driver.configure.signaling.ueAssistance.nradio.get_value()
```

Configures UE assistance requests for power saving and handling of overheating. This command combines the other configuration commands.

return
structure: for return value, see the help for ValueStruct structure arguments.

set_value(*value: ValueStruct*) → None

```
# SCPI: [CONFigure]:SIGNaling:UEASsistance:NRADio
structure = driver.configure.signaling.ueAssistance.nradio.ValueStruct()
structure.Dbr_Enable: bool = False
structure.Dbr_Prohibit_Timer: enums.ProhibitTimer = enums.ProhibitTimer.INF
structure.Oass_Enable: bool = False
structure.Oass_Prohibit_Timer: enums.ProhibitTimer = enums.ProhibitTimer.INF
structure.Drxp_Enable: bool = False
structure.Drxp_Prohibit_Timer: enums.ProhibitTimer = enums.ProhibitTimer.INF
structure.Mbwp_Enable: bool = False
structure.Mbwp_Prohibit_Timer: enums.ProhibitTimer = enums.ProhibitTimer.INF
structure.Mccp_Enable: bool = False
structure.Mccp_Prohibit_Timer: enums.ProhibitTimer = enums.ProhibitTimer.INF
structure.Mml_Enable: bool = False
structure.Mml_Prohibit_Timer: enums.ProhibitTimer = enums.ProhibitTimer.INF
structure.Msof_Enable: bool = False
structure.Msof_Prohibit_Timer: enums.ProhibitTimer = enums.ProhibitTimer.INF
structure.Relp_Enable: bool = False
structure.Relp_Prohibit_Timer: enums.ProhibitTimer = enums.ProhibitTimer.INF
driver.configure.signaling.ueAssistance.nradio.set_value(value = structure)
```

Configures UE assistance requests for power saving and handling of overheating. This command combines the other configuration commands.

param value
see the help for ValueStruct structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.signaling.ueAssistance.nradio.clone()
```

Subgroups

6.3.4.16.1.1 DbReport

SCPI Command :

```
[CONFigure]:SIGNaling:UEASsistance:NRADio:DBReport
```

class DbReportCls

DbReport commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class DbReportStruct

Response structure. Fields:

- Enable: bool: Enables/disables transmitting the parameter ‘delayBudgetReportingConfig’ in the IE ‘OtherConfig’.

- Prohibit_Timer: enums.ProhibitTimer: Signaled 'delayBudgetReportingProhibitTimer'. Sn: n ms SnDm: n.m ms

get() → DbReportStruct

```
# SCPI: [CONFigure]:SIGNaling:UEASsistance:NRADio:DBReport
value: DbReportStruct = driver.configure.signaling.ueAssistance.nradio.dbReport.
↪get()
```

Configures requests for delay budget reports.

return
structure: for return value, see the help for DbReportStruct structure arguments.

set(*enable: bool*, *prohibit_timer: ProhibitTimer = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:UEASsistance:NRADio:DBReport
driver.configure.signaling.ueAssistance.nradio.dbReport.set(enable = False,␣
↪prohibit_timer = enums.ProhibitTimer.INF)
```

Configures requests for delay budget reports.

param enable
Enables/disables transmitting the parameter 'delayBudgetReportingConfig' in the IE 'OtherConfig'.

param prohibit_timer
Signaled 'delayBudgetReportingProhibitTimer'. Sn: n ms SnDm: n.m ms

6.3.4.16.1.2 DrxPref

SCPI Command :

```
[CONFigure]:SIGNaling:UEASsistance:NRADio:DRXPref
```

class DrxPrefCls

DrxPref commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class DrxPrefStruct

Response structure. Fields:

- Enable: bool: Enables/disables transmitting the parameter 'DRX-PreferenceConfig-r16' in the IE 'OtherConfig'.
- Prohibit_Timer: enums.ProhibitTimer: Signaled 'drx-PreferenceProhibitTimer-r16'. Sn: n ms SnDm: n.m ms

get() → DrxPrefStruct

```
# SCPI: [CONFigure]:SIGNaling:UEASsistance:NRADio:DRXPref
value: DrxPrefStruct = driver.configure.signaling.ueAssistance.nradio.drxPref.
↪get()
```

Configures requests for DRX preferences.

return
structure: for return value, see the help for DrxPrefStruct structure arguments.

set(*enable: bool*, *prohibit_timer: ProhibitTimer = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:UEASsistance:NRADio:DRXPref
driver.configure.signaling.ueAssistance.nradio.drxPref.set(enable = False,␣
↪prohibit_timer = enums.ProhibitTimer.INF)
```

Configures requests for DRX preferences.

param enable
Enables/disables transmitting the parameter 'DRX-PreferenceConfig-r16' in the IE 'OtherConfig'.

param prohibit_timer
Signaled 'drx-PreferenceProhibitTimer-r16'. Sn: n ms SnDm: n.m ms

6.3.4.16.1.3 MbwPref

SCPI Command :

```
[CONFigure]:SIGNaling:UEASsistance:NRADio:MBWPref
```

class MbwPrefCls

MbwPref commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class MbwPrefStruct

Response structure. Fields:

- Enable: bool: Enables/disables transmitting the parameter 'MaxBW-PreferenceConfig-r16' in the IE 'OtherConfig'.
- Prohibit_Timer: enums.ProhibitTimer: Signaled 'maxBW-PreferenceProhibitTimer-r16'. Sn: n ms SnDm: n.m ms

get() → MbwPrefStruct

```
# SCPI: [CONFigure]:SIGNaling:UEASsistance:NRADio:MBWPref
value: MbwPrefStruct = driver.configure.signaling.ueAssistance.nradio.mbwPref.
↪get()
```

Configures requests for the preferred maximum aggregated bandwidth.

return
structure: for return value, see the help for MbwPrefStruct structure arguments.

set(*enable: bool*, *prohibit_timer: ProhibitTimer = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:UEASsistance:NRADio:MBWPref
driver.configure.signaling.ueAssistance.nradio.mbwPref.set(enable = False,␣
↪prohibit_timer = enums.ProhibitTimer.INF)
```

Configures requests for the preferred maximum aggregated bandwidth.

param enable
Enables/disables transmitting the parameter 'MaxBW-PreferenceConfig-r16' in the IE 'OtherConfig'.

param prohibit_timer
Signaled 'maxBW-PreferenceProhibitTimer-r16'. Sn: n ms SnDm: n.m ms

6.3.4.16.1.4 MccPref

SCPI Command :

```
[CONFigure]:SIGNaling:UEASsistance:NRADio:MCCPref
```

class MccPrefCls

MccPref commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class MccPrefStruct

Response structure. Fields:

- Enable: bool: Enables/disables transmitting the parameter 'MaxCC-PreferenceConfig-r16' in the IE 'OtherConfig'.
- Prohibit_Timer: enums.ProhibitTimer: Signaled 'maxCC-PreferenceProhibitTimer-r16'. Sn: n ms SnDm: n.m ms

get() → MccPrefStruct

```
# SCPI: [CONFigure]:SIGNaling:UEASsistance:NRADio:MCCPref
value: MccPrefStruct = driver.configure.signaling.ueAssistance.nradio.mccPref.
↪get()
```

Configures requests for the preferred maximum number of carriers.

return
structure: for return value, see the help for MccPrefStruct structure arguments.

set(*enable: bool*, *prohibit_timer: ProhibitTimer = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:UEASsistance:NRADio:MCCPref
driver.configure.signaling.ueAssistance.nradio.mccPref.set(enable = False,␣
↪prohibit_timer = enums.ProhibitTimer.INF)
```

Configures requests for the preferred maximum number of carriers.

param enable
Enables/disables transmitting the parameter 'MaxCC-PreferenceConfig-r16' in the IE 'OtherConfig'.

param prohibit_timer
Signaled 'maxCC-PreferenceProhibitTimer-r16'. Sn: n ms SnDm: n.m ms

6.3.4.16.1.5 MmLayer

SCPI Command :

```
[CONFigure]:SIGNaling:UEASsistance:NRADio:MMLayer
```

class MmLayerCls

MmLayer commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class MmLayerStruct

Response structure. Fields:

- Enable: bool: Enables/disables transmitting the parameter 'MaxMIMO-LayerPreferenceConfig-r16' in the IE 'OtherConfig'.
- Prohibit_Timer: enums.ProhibitTimer: Signaled 'maxMIMO-LayerPreferenceProhibitTimer-r16'. Sn: n ms SnDm: n.m ms

get() → MmLayerStruct

```
# SCPI: [CONFigure]:SIGNaling:UEASsistance:NRADio:MMLayer
value: MmLayerStruct = driver.configure.signaling.ueAssistance.nradio.mmLayer.
↪get()
```

Configures requests for the preferred maximum number of MIMO layers.

return
structure: for return value, see the help for MmLayerStruct structure arguments.

set(*enable: bool*, *prohibit_timer: ProhibitTimer = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:UEASsistance:NRADio:MMLayer
driver.configure.signaling.ueAssistance.nradio.mmLayer.set(enable = False,␣
↪prohibit_timer = enums.ProhibitTimer.INF)
```

Configures requests for the preferred maximum number of MIMO layers.

param enable
Enables/disables transmitting the parameter 'MaxMIMO-LayerPreferenceConfig-r16' in the IE 'OtherConfig'.

param prohibit_timer
Signaled 'maxMIMO-LayerPreferenceProhibitTimer-r16'. Sn: n ms SnDm: n.m ms

6.3.4.16.1.6 MsOffset

SCPI Command :

```
[CONFigure]:SIGNaling:UEASsistance:NRADio:MSOFfset
```

class MsOffsetCls

MsOffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class MsOffsetStruct

Response structure. Fields:

- Enable: bool: Enables/disables transmitting the parameter 'MinSchedulingOffsetPreferenceConfig-r16' in the IE 'OtherConfig'.
- Prohibit_Timer: enums.ProhibitTimer: Signaled 'minSchedulingOffsetPreferenceProhibitTimer-r16'. Sn: n ms SnDm: n.m ms

get() → MsOffsetStruct

```
# SCPI: [CONFigure]:SIGNaling:UEASsistance:NRADio:MSOFfset
value: MsOffsetStruct = driver.configure.signaling.ueAssistance.nradio.msOffset.
↪get()
```

Configures requests for the preferred minimum scheduling offset.

return
structure: for return value, see the help for MsOffsetStruct structure arguments.

set(*enable: bool*, *prohibit_timer: ProhibitTimer = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:UEASsistance:NRADio:MSOFfset
driver.configure.signaling.ueAssistance.nradio.msOffset.set(enable = False,␣
↪prohibit_timer = enums.ProhibitTimer.INF)
```

Configures requests for the preferred minimum scheduling offset.

param enable
Enables/disables transmitting the parameter 'MinSchedulingOffsetPreferenceConfig-r16' in the IE 'OtherConfig'.

param prohibit_timer
Signaled 'minSchedulingOffsetPreferenceProhibitTimer-r16'. Sn: n ms SnDm: n.m ms

6.3.4.16.1.7 Oassistance

SCPI Command :

```
[CONFigure]:SIGNaling:UEASsistance:NRADio:OASSistance
```

class OassistanceCls

Oassistance commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class OassistanceStruct

Response structure. Fields:

- Enable: bool: Enables/disables transmitting the parameter 'OverheatingAssistanceConfig' in the IE 'OtherConfig'.
- Prohibit_Timer: enums.ProhibitTimer: Signaled 'overheatingIndicationProhibitTimer'. Sn: n ms SnDm: n.m ms

get() → OassistanceStruct

```
# SCPI: [CONFigure]:SIGNaling:UEASsistance:NRADio:OASSistance
value: OassistanceStruct = driver.configure.signaling.ueAssistance.nradio.
↪oassistance.get()
```

Configures requests for overheating assistance information.

return
structure: for return value, see the help for OassistanceStruct structure arguments.

set(*enable: bool*, *prohibit_timer: ProhibitTimer = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:UEASsistance:NRADio:OASSistance
driver.configure.signaling.ueAssistance.nradio.oassistance.set(enable = False,␣
↪prohibit_timer = enums.ProhibitTimer.INF)
```

Configures requests for overheating assistance information.

param enable
Enables/disables transmitting the parameter 'OverheatingAssistanceConfig' in the IE 'OtherConfig'.

param prohibit_timer
Signaled 'overheatingIndicationProhibitTimer'. Sn: n ms SnDm: n.m ms

6.3.4.16.1.8 RelPref

SCPI Command :

```
[CONFigure]:SIGNaling:UEASsistance:NRADio:RELPref
```

class RelPrefCls

RelPref commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class RelPrefStruct

Response structure. Fields:

- Enable: bool: Enables/disables transmitting the parameter 'ReleasePreferenceConfig-r16' in the IE 'OtherConfig'.
- Prohibit_Timer: enums.ProhibitTimer: Signaled 'releasePreferenceProhibitTimer-r16'. Sn: n ms SnDm: n.m ms

get() → RelPrefStruct

```
# SCPI: [CONFigure]:SIGNaling:UEASsistance:NRADio:RELPref
value: RelPrefStruct = driver.configure.signaling.ueAssistance.nradio.relPref.
↪get()
```

Configures requests for connection release preference information.

return
structure: for return value, see the help for RelPrefStruct structure arguments.

set(*enable: bool*, *prohibit_timer: ProhibitTimer = None*) → None

```
# SCPI: [CONFigure]:SIGNaling:UEASsistance:NRADio:RELPref
driver.configure.signaling.ueAssistance.nradio.relPref.set(enable = False,␣
↪prohibit_timer = enums.ProhibitTimer.INF)
```

Configures requests for connection release preference information.

param enable
Enables/disables transmitting the parameter 'ReleasePreferenceConfig-r16' in the IE 'OtherConfig'.

param prohibit_timer
Signaled 'releasePreferenceProhibitTimer-r16'. Sn: n ms SnDm: n.m ms

6.3.5 Wlan

class WlanCls

Wlan commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlan.clone()
```

Subgroups

6.3.5.1 Measurement<MeasInstance>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.configure.wlan.measurement.repcap_measInstance_get()
driver.configure.wlan.measurement.repcap_measInstance_set(repcap.MeasInstance.Nr1)
```

class MeasurementCls

Measurement commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability: MeasInstance, default value after init: MeasInstance.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlan.measurement.clone()
```

Subgroups

6.3.5.1.1 Network

class NetworkCls

Network commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.wlan.measurement.network.clone()
```

Subgroups

6.3.5.1.1.1 Cell

SCPI Command :

```
[CONFigure]:WLAN:MEASurement<Instance>:NETWork:CELL
```

class CellCls

Cell commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class CellStruct

Response structure. Fields:

- Cell_Name: str: No parameter help available
- Used_Ul: List[int]: No parameter help available

get(*measInstance=MeasInstance.Default*) → CellStruct

```
# SCPI: [CONFigure]:WLAN:MEASurement<Instance>:NETWork:CELL
value: CellStruct = driver.configure.wlan.measurement.network.cell.
↪get(measInstance = repcap.MeasInstance.Default)
```

No command help available

param measInstance
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Measurement')

return
structure: for return value, see the help for CellStruct structure arguments.

set(*cell_name: str*, *used_ul: List[int] = None*, *measInstance=MeasInstance.Default*) → None

```
# SCPI: [CONFigure]:WLAN:MEASurement<Instance>:NETWork:CELL
driver.configure.wlan.measurement.network.cell.set(cell_name = 'abc', used_ul =␣
↪[1, 2, 3], measInstance = repcap.MeasInstance.Default)
```

No command help available

param cell_name
No help available

param used_ul
No help available

param measInstance
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Measurement')

6.4 Create

class CreateCls

Create commands group definition. 19 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.create.clone()
```

Subgroups

6.4.1 Signaling

SCPI Commands :

```
CREate:SIGNaling:CMAS
CREate:SIGNaling:RFCHannel
CREate:SIGNaling:FADing
```

class SignalingCls

Signaling commands group definition. 19 total commands, 6 Subgroups, 3 group commands

set_cmas(*network_scope: str*) → None

```
# SCPI: CREate:SIGNaling:CMAS
driver.create.signaling.set_cmas(network_scope = 'abc')
```

Creates a CMAS service for all cells in a certain <NetworkScope>. Use this network scope in the other CMAS commands.

param network_scope
Name of a PLMN or a tracking area or a cell

set_fading(*cell_name: str*) → None

```
# SCPI: CREate:SIGNaling:FADing
driver.create.signaling.set_fading(cell_name = 'abc')
```

Allows fading and reserves the required resources. Send this command before switching to live mode.

param cell_name
No help available

set_rf_channel(*cell_name: str*) → None

```
# SCPI: CREate:SIGNaling:RFCHannel
driver.create.signaling.set_rf_channel(cell_name = 'abc')
```

No command help available

param cell_name
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.create.signaling.clone()
```

Subgroups

6.4.1.1 Awgn

SCPI Command :

```
CREate:SIGNaling:AWGN:ADVanced
```

class AwgnCls

Awgn commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set_advanced(*cell_name: str*) → None

```
# SCPI: CREate:SIGNaling:AWGN:ADVanced
driver.create.signaling.awgn.set_advanced(cell_name = 'abc')
```

No command help available

param cell_name
No help available

6.4.1.2 Ccopy

SCPI Command :

```
CREate:SIGNaling:CCOPy
```

class CcopyCls

Ccopy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*cell_name: str*, *no_copies: int*, *continuous: bool*) → None

```
# SCPI: CREate:SIGNaling:CCOPy
driver.create.signaling.ccopy.set(cell_name = 'abc', no_copies = 1, continuous
↪= False)
```

Copies a cell.

param cell_name
Name of the source cell.

param no_copies
Number of cell copies to be created.

param continuous

- ON: Places the cell copies above the source cell in the same frequency band. Configures the frequencies of the cell copies for intraband contiguous carrier aggregation.
- OFF: Copies the frequency settings of the source cell.

6.4.1.3 Etws

SCPI Commands :

```
CREate:SIGNaling:ETWS:SECondary
CREate:SIGNaling:ETWS
```

class EtwsCls

Etws commands group definition. 2 total commands, 0 Subgroups, 2 group commands

set_secondary(*network_scope: str*) → None

```
# SCPI: CREate:SIGNaling:ETWS:SECondary
driver.create.signaling.etws.set_secondary(network_scope = 'abc')
```

Creates an ETWS secondary service for all cells in a certain <NetworkScope>. Use this network scope in the other ETWS secondary commands.

param network_scope
Name of a PLMN or a tracking area or a cell

set_value(*network_scope: str*) → None

```
# SCPI: CREate:SIGNaling:ETWS
driver.create.signaling.etws.set_value(network_scope = 'abc')
```

Creates an ETWS primary service for all cells in a certain <NetworkScope>. Use this network scope in the other ETWS primary commands.

param network_scope
Name of a PLMN or a tracking area or a cell

6.4.1.4 Lte

SCPI Command :

```
CREate:SIGNaling:LTE:CGRoup
```

class LteCls

Lte commands group definition. 3 total commands, 2 Subgroups, 1 group commands

set_cgroup(*cell_group_name: str*) → None

```
# SCPI: CREate:SIGNaling:LTE:CGRoup
driver.create.signaling.lte.set_cgroup(cell_group_name = 'abc')
```

Creates an LTE or NR cell group. Assign a unique name to each named object within the test environment. Assigning an already used name can be rejected with an error message, even if the other object has not the same type as the new object.

param cell_group_name
Assigns a name to the cell group. The string is used in other commands to select this cell group.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.create.signaling.lte.clone()
```

Subgroups

6.4.1.4.1 Cell

SCPI Command :

```
CREate:SIGNaling:LTE:CELL
```

class CellCls

Cell commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*cell_name: str*, *preferred_netw: PreferredNetw = None*, *physical_cell_id: float = None*) → None

```
# SCPI: CREate:SIGNaling:LTE:CELL
driver.create.signaling.lte.cell.set(cell_name = 'abc', preferred_netw = enums.
↪PreferredNetw.AUTO, physical_cell_id = 1.0)
```

Creates a physical LTE cell (a cell that can be switched on) . Assign a unique name to each named object within the test environment. Assigning an already used name can be rejected with an error message, even if the other object has not the same type as the new object.

param cell_name
Assigns a name to the cell. The string is used in other commands to select this cell.

param preferred_netw
No help available

param physical_cell_id
No help available

6.4.1.4.2 Vcell

SCPI Command :

```
CREate:SIGNaling:LTE:VCELl
```

class VcellCls

Vcell commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*cell_name: str*, *fbi: float = None*, *channel: float = None*) → None

```
# SCPI: CREate:SIGNaling:LTE:VCELl
driver.create.signaling.lte.vcell.set(cell_name = 'abc', fbi = 1.0, channel = 1.
↪0)
```

Creates a virtual LTE cell. A virtual cell cannot be switched on. Assign a unique name to each named object within the test environment. Assigning an already used name can be rejected with an error message, even if the other object has not the same type as the new object.

param cell_name
Assigns a name to the cell. The string is used in other commands to select this cell.

param fbi
Frequency band indicator.

param channel
Channel number (EARFCN) .

6.4.1.5 Nradio

SCPI Command :

```
CREate:SIGNaling:NRADio:CGRoup
```

class NradioCls

Nradio commands group definition. 3 total commands, 2 Subgroups, 1 group commands

set_cgroup(*cell_group_name: str*) → None

```
# SCPI: CREate:SIGNaling:NRADio:CGRoup
driver.create.signaling.nradio.set_cgroup(cell_group_name = 'abc')
```

Creates an LTE or NR cell group. Assign a unique name to each named object within the test environment. Assigning an already used name can be rejected with an error message, even if the other object has not the same type as the new object.

param cell_group_name
Assigns a name to the cell group. The string is used in other commands to select this cell group.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.create.signaling.nradio.clone()
```

Subgroups

6.4.1.5.1 Cell

SCPI Command :

```
CREate:SIGNaling:NRADio:CELL
```

class CellCls

Cell commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*cell_name: str*, *physical_cell_id: int = None*) → None

```
# SCPI: CREate:SIGNaling:NRADio:CELL
driver.create.signaling.nradio.cell.set(cell_name = 'abc', physical_cell_id = 1)
```

The command creates a physical NR cell (a cell that can be switched on) . Assign a unique name to each named object within the test environment. Assigning an already used name can be rejected with an error message, even if the other object has not the same type as the new object.

param cell_name
Assigns a name to the cell. The string is used in other commands to select this cell.

param physical_cell_id
No help available

6.4.1.5.2 Vcell

SCPI Command :

```
CREate:SIGNaling:NRADio:VCELl
```

class VcellCls

Vcell commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*cell_name: str*, *fbi: int = None*, *channel: int = None*) → None

```
# SCPI: CREate:SIGNaling:NRADio:VCELl
driver.create.signaling.nradio.vcell.set(cell_name = 'abc', fbi = 1, channel =␣
↪1)
```

Creates a virtual NR cell. A virtual cell cannot be switched on. Assign a unique name to each named object within the test environment. Assigning an already used name can be rejected with an error message, even if the other object has not the same type as the new object.

param cell_name
Assigns a name to the cell. The string is used in other commands to select this cell.

param fbi
Frequency band indicator.

param channel
Channel number (NR ARFCN) .

6.4.1.6 Topology

class TopologyCls

Topology commands group definition. 6 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.create.signaling.topology.clone()
```

Subgroups

6.4.1.6.1 Cnetwork

SCPI Command :

```
CREate:SIGNaling:TOPology:CNETwork
```

class CnetworkCls

Cnetwork commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: CREate:SIGNaling:TOPology:CNETwork
driver.create.signaling.topology.cnetwork.set()
```

Creates the core network. This action can take some minutes.

set_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: CREate:SIGNaling:TOPology:CNETwork
driver.create.signaling.topology.cnetwork.set_with_opc()
```

Creates the core network. This action can take some minutes.

Same as set, but waits for the operation to complete before continuing further. Use the RsCMX_Signaling.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

6.4.1.6.2 Eps

SCPI Commands :

```
CREate:SIGNaling:TOPology:EPS
CREate:SIGNaling:TOPology:EPS:BEARer
```

class EpsCls

Eps commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class BearerStruct

Structure for setting input parameters. Contains optional set arguments. Fields:

- Linked_Bearer_Id: int: ID of the default bearer to which the dedicated bearer is linked. To get a list of all default bearer IDs, see [CMDLINKRESOLVED Catalog.Signaling.Lte.Ue.Dbearer#get_CMDLINKRESOLVED].
- Qci: enums.Qi: Optional setting parameter. Value of the quality of service class identifier. Values defined in 3GPP TS 23.203, table 6.1.7. The GUI shows the designation of each value.
- Rlc_Mode: enums.RlcMode: Optional setting parameter. ACK: acknowledged UACK: unacknowledged
- Max_Dl_Bitrate: int: Optional setting parameter. Maximum DL bit rate allowed in the network.
- Max_Ul_Bitrate: int: Optional setting parameter. Maximum UL bit rate allowed in the network.
- Gtd_Dl_Bitrate: int: Optional setting parameter. DL bit rate guaranteed by the network for the bearer.
- Gtd_Ul_Bitrate: int: Optional setting parameter. UL bit rate guaranteed by the network for the bearer.
- Local_Port_Lower: int: Optional setting parameter. The minimum of the local port range in the traffic flow template.
- Local_Port_Upper: int: Optional setting parameter. The maximum of the local port range in the traffic flow template.
- Remote_Port_Lower: int: Optional setting parameter. The minimum of the remote port range in the traffic flow template.
- Remote_Port_Upper: int: Optional setting parameter. The maximum of the remote port range in the traffic flow template.

set(*name_ta: str*, *name_plmn: str*, *ta_code: int = None*, *time_3412: float = None*) → None

```
# SCPI: CREate:SIGNaling:TOPology:EPS
driver.create.signaling.topology.eps.set(name_ta = 'abc', name_plmn = 'abc', ta_
↪code = 1, time_3412 = 1.0)
```

Creates an EPS tracking area in a selected PLMN and optionally defines tracking area settings. Assign a unique name to each named object within the test environment. Assigning an already used name can be rejected with an error message, even if the other object has not the same type as the new object.

param name_ta
Assigns a name to the tracking area. The string is used in other commands to select this tracking area.

param name_plmn
PLMN containing the tracking area.

param ta_code
Tracking area code (TAC) .

param time_3412
No effect - for future use.

`set_bearer`(*value: BearerStruct*) → None

```
# SCPI: CREate:SIGNaling:TOPology:EPS:BEARer
structure = driver.create.signaling.topology.eps.BearerStruct()
structure.Linked_Bearer_Id: int = 1
structure.Qci: enums.Qi = enums.Qi.Q1
structure.Rlc_Mode: enums.RlcMode = enums.RlcMode.ACK
structure.Max_Dl_Bitrate: int = 1
structure.Max_Ul_Bitrate: int = 1
structure.Gtd_Dl_Bitrate: int = 1
structure.Gtd_Ul_Bitrate: int = 1
structure.Local_Port_Lower: int = 1
structure.Local_Port_Upper: int = 1
structure.Remote_Port_Lower: int = 1
structure.Remote_Port_Upper: int = 1
driver.create.signaling.topology.eps.set_bearer(value = structure)
```

Establishes a dedicated bearer.

param value
see the help for BearerStruct structure arguments.

6.4.1.6.3 Fgs

SCPI Command :

```
CREate:SIGNaling:TOPology:FGS
```

`class FgsCls`

Fgs commands group definition. 2 total commands, 1 Subgroups, 1 group commands

`set`(*name_ta_5_g: str*, *name_plmn: str*, *ta_code: int = None*) → None

```
# SCPI: CREate:SIGNaling:TOPology:FGS
driver.create.signaling.topology.fgs.set(name_ta_5_g = 'abc', name_plmn = 'abc',
↪ ta_code = 1)
```

Creates a 5GS tracking area in a selected PLMN and optionally defines the TAC. Assign a unique name to each named object within the test environment. Assigning an already used name can be rejected with an error message, even if the other object has not the same type as the new object.

param name_ta_5_g
Assigns a name to the tracking area. The string is used in other commands to select this tracking area.

param name_plmn
PLMN containing the tracking area.

param ta_code
Tracking area code (TAC) .

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.create.signaling.topology.fgs.clone()
```

Subgroups

6.4.1.6.3.1 Ue

class UeCls

Ue commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.create.signaling.topology.fgs.ue.clone()
```

Subgroups

6.4.1.6.3.2 Pdu

class PduCls

Pdu commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.create.signaling.topology.fgs.ue.pdu.clone()
```

Subgroups

6.4.1.6.3.3 QosFlow

SCPI Command :

```
CREate:SIGNaling:TOPology:FGS:UE:PDU:QOSFlow
```

class QosFlowCls

QosFlow commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class SetStruct

Structure for setting input parameters. Contains optional setting parameters. Fields:

- Ue_Id: str: Optional setting parameter. For future use. Enter any value.
- Pdu_Session_Id: int: Optional setting parameter. ID of the existing PDU session to which the QoS flow is added.

- Qi: enums.Qi: Optional setting parameter. 5G quality of service identifier (5QI) .
- Max_Dl_Bitrate: int: Optional setting parameter. Maximum flow bit rate (MFBR) for the DL.
- Max_Dl_Unit: enums.ItRateUnit: Optional setting parameter. Unit for 4_MaxDLBitrate. Kn, Mn, Gn, Tn, Pn = n kbit/s, Mbit/s, Gbit/s, Tbit/s, Pbit/s
- Max_Ul_Bitrate: int: Optional setting parameter. Maximum flow bit rate (MFBR) for the UL.
- Max_Ul_Unit: enums.ItRateUnit: Optional setting parameter. Unit for 6_MaxULBitrate.
- Flow_Control: enums.FlowControl: Optional setting parameter. GUARanteed: GBR QoS flow NGUaranteed: non-GBR QoS flow
- Dl_Bitrate: int: Optional setting parameter. Guaranteed flow bit rate (GFBR) for the DL, only for GBR QoS flows.
- Dl_Unit: enums.ItRateUnit: Optional setting parameter. Unit for 9_DLBitrate, only for GBR QoS flows.
- Ul_Bitrate: int: Optional setting parameter. Guaranteed flow bit rate (GFBR) for the UL, only for GBR QoS flows.
- Ul_Unit: enums.ItRateUnit: Optional setting parameter. Unit for 11_ULBitrate, only for GBR QoS flows.
- Averaging_Window: int or bool: Optional setting parameter. Duration over which the bit rates GFBR and MFBR are calculated for GBR QoS flows. OFF omits the parameter in the QoS flow description.
- Rlc_Mode: enums.RlcMode: Optional setting parameter. RLC mode ACK: acknowledged UACK: unacknowledged

set(*structure: SetStruct*) → None

```
# SCPI: CREate:SIGNaling:TOPology:FGS:UE:PDU:QOSFlow
structure = driver.create.signaling.topology.fgs.ue.pdu.qosFlow.SetStruct()
structure.Ue_Id: str = 'abc'
structure.Pdu_Session_Id: int = 1
structure.Qi: enums.Qi = enums.Qi.Q1
structure.Max_Dl_Bitrate: int = 1
structure.Max_Dl_Unit: enums.ItRateUnit = enums.ItRateUnit.G1
structure.Max_Ul_Bitrate: int = 1
structure.Max_Ul_Unit: enums.ItRateUnit = enums.ItRateUnit.G1
structure.Flow_Control: enums.FlowControl = enums.FlowControl.GUARanteed
structure.Dl_Bitrate: int = 1
structure.Dl_Unit: enums.ItRateUnit = enums.ItRateUnit.G1
structure.Ul_Bitrate: int = 1
structure.Ul_Unit: enums.ItRateUnit = enums.ItRateUnit.G1
structure.Averaging_Window: int or bool = 1
structure.Rlc_Mode: enums.RlcMode = enums.RlcMode.ACK
driver.create.signaling.topology.fgs.ue.pdu.qosFlow.set(structure)
```

Adds a QoS flow to an existing PDU session.

param structure
for set value, see the help for SetStruct structure arguments.

6.4.1.6.4 Plmn

SCPI Command :

```
CREate:SIGNaling:TOPology:PLMN
```

class PlmnCls

Plmn commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*name_plmn: str*, *mcc: str = None*, *mnc: str = None*) → None

```
# SCPI: CREate:SIGNaling:TOPology:PLMN
driver.create.signaling.topology.plmn.set(name_plmn = 'abc', mcc = 'abc', mnc =
↪'abc')
```

Creates a PLMN and optionally defines the MCC and MNC of the PLMN. Assign a unique name to each named object within the test environment. Assigning an already used name can be rejected with an error message, even if the other object has not the same type as the new object.

param name_plmn
Assigns a name to the PLMN. The string is used in other commands to select this PLMN.

param mcc
No help available

param mnc
No help available

6.5 Diagnostic

class DiagnosticCls

Diagnostic commands group definition. 19 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.diagnostic.clone()
```

Subgroups

6.5.1 Signaling

class SignalingCls

Signaling commands group definition. 19 total commands, 10 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.diagnostic.signaling.clone()
```

Subgroups

6.5.1.1 Dapi

SCPI Command :

```
DIAGnostic:SIGNaling:DAPI:TOUT
```

class DapiCls

Dapi commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_timeout() → int

```
# SCPI: DIAGnostic:SIGNaling:DAPI:TOUT
value: int = driver.diagnostic.signaling.dapi.get_timeout()
```

No command help available

return
timeout: No help available

set_timeout(*timeout: int*) → None

```
# SCPI: DIAGnostic:SIGNaling:DAPI:TOUT
driver.diagnostic.signaling.dapi.set_timeout(timeout = 1)
```

No command help available

param timeout
No help available

6.5.1.2 Eps

class EpsCls

Eps commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.diagnostic.signaling.eps.clone()
```

Subgroups

6.5.1.2.1 Logging

class LoggingCls

Logging commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.diagnostic.signaling.eps.logging.clone()
```

Subgroups

6.5.1.2.1.1 Uplane

SCPI Commands :

```
DIAGnostic:SIGNaling:EPS:LOGGing:UPLane:UL
DIAGnostic:SIGNaling:EPS:LOGGing:UPLane:DL
```

class UplaneCls

Uplane commands group definition. 2 total commands, 0 Subgroups, 2 group commands

get_downlink() → LogLevel

```
# SCPI: DIAGnostic:SIGNaling:EPS:LOGGing:UPLane:DL
value: enums.LogLevel = driver.diagnostic.signaling.eps.logging.uplane.get_
↪downlink()
```

No command help available

return
log_level: No help available

get_uplink() → LogLevel

```
# SCPI: DIAGnostic:SIGNaling:EPS:LOGGing:UPLane:UL
value: enums.LogLevel = driver.diagnostic.signaling.eps.logging.uplane.get_
↪uplink()
```

No command help available

return
log_level: No help available

set_downlink(*log_level: LogLevel*) → None

```
# SCPI: DIAGnostic:SIGNaling:EPS:LOGGing:UPLane:DL
driver.diagnostic.signaling.eps.logging.uplane.set_downlink(log_level = enums.
↪LogLevel.BRIef)
```

No command help available

param log_level
No help available

set_uplink(*log_level: LogLevel*) → None

```
# SCPI: DIAGnostic:SIGNaling:EPS:LOGGing:UPLane:UL
driver.diagnostic.signaling.eps.logging.uplane.set_uplink(log_level = enums.
↪LogLevel.BRIef)
```

No command help available

param log_level
No help available

6.5.1.3 Fgs

class FgsCls

Fgs commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.diagnostic.signaling.fgs.clone()
```

Subgroups

6.5.1.3.1 Logging

class LoggingCls

Logging commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.diagnostic.signaling.fgs.logging.clone()
```

Subgroups

6.5.1.3.1.1 Uplane

SCPI Commands :

```
DIAGnostic:SIGNaling:FGS:LOGGing:UPLane:UL
DIAGnostic:SIGNaling:FGS:LOGGing:UPLane:DL
```

class UplaneCls

Uplane commands group definition. 2 total commands, 0 Subgroups, 2 group commands

get_downlink() → LogLevel

```
# SCPI: DIAGnostic:SIGNaling:FGS:LOGGing:UPLane:DL
value: enums.LogLevel = driver.diagnostic.signaling.fgs.logging.uplane.get_
↪downlink()
```

No command help available

return
log_level: No help available

get_uplink() → LogLevel

```
# SCPI: DIAGnostic:SIGNaling:FGS:LOGGing:UPLane:UL
value: enums.LogLevel = driver.diagnostic.signaling.fgs.logging.uplane.get_
↪uplink()
```

No command help available

return
log_level: No help available

set_downlink(*log_level: LogLevel*) → None

```
# SCPI: DIAGnostic:SIGNaling:FGS:LOGGing:UPLane:DL
driver.diagnostic.signaling.fgs.logging.uplane.set_downlink(log_level = enums.
↪LogLevel.BRIef)
```

No command help available

param log_level
No help available

set_uplink(*log_level: LogLevel*) → None

```
# SCPI: DIAGnostic:SIGNaling:FGS:LOGGing:UPLane:UL
driver.diagnostic.signaling.fgs.logging.uplane.set_uplink(log_level = enums.
↪LogLevel.BRIef)
```

No command help available

param log_level
No help available

6.5.1.4 Logging

class LoggingCls

Logging commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.diagnostic.signaling.logging.clone()
```

Subgroups

6.5.1.4.1 Uplane

SCPI Commands :

```
DIAGnostic:SIGNaling:LOGGing:UPLane:UL
DIAGnostic:SIGNaling:LOGGing:UPLane:DL
```

class UplaneCls

Uplane commands group definition. 2 total commands, 0 Subgroups, 2 group commands

get_downlink() → LogLevel

```
# SCPI: DIAGnostic:SIGNaling:LOGGing:UPLane:DL
value: enums.LogLevel = driver.diagnostic.signaling.logging.uplane.get_
↪downlink()
```

No command help available

return
log_level: No help available

get_uplink() → LogLevel

```
# SCPI: DIAGnostic:SIGNaling:LOGGing:UPLane:UL
value: enums.LogLevel = driver.diagnostic.signaling.logging.uplane.get_uplink()
```

No command help available

return
log_level: No help available

set_downlink(*log_level: LogLevel*) → None

```
# SCPI: DIAGnostic:SIGNaling:LOGGing:UPLane:DL
driver.diagnostic.signaling.logging.uplane.set_downlink(log_level = enums.
↪LogLevel.BRIef)
```

No command help available

param log_level
No help available

set_uplink(*log_level: LogLevel*) → None

```
# SCPI: DIAGnostic:SIGNaling:LOGGing:UPLane:UL
driver.diagnostic.signaling.logging.uplane.set_uplink(log_level = enums.
↪LogLevel.BRIef)
```

No command help available

param log_level
No help available

6.5.1.5 Lte

class LteCls

Lte commands group definition. 3 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.diagnostic.signaling.lte.clone()
```

Subgroups

6.5.1.5.1 Cell

class CellCls

Cell commands group definition. 3 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.diagnostic.signaling.lte.cell.clone()
```

Subgroups

6.5.1.5.1.1 Logging

class LoggingCls

Logging commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.diagnostic.signaling.lte.cell.logging.clone()
```

Subgroups

6.5.1.5.1.2 Mac

SCPI Command :

```
DIAGnostic:SIGNaling:LTE:CELL:LOGGing:MAC
```

class MacCls

Mac commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class MacStruct

Response structure. Fields:

- Enable: bool: No parameter help available
- Log_Type: enums.LogType: No parameter help available
- Payload: int: No parameter help available

get() → MacStruct

```
# SCPI: DIAGnostic:SIGNaling:LTE:CELL:LOGGing:MAC
value: MacStruct = driver.diagnostic.signaling.lte.cell.logging.mac.get()
```

No command help available

return
structure: for return value, see the help for MacStruct structure arguments.

set(*enable: bool*, *log_type: LogType*, *payload: int = None*) → None

```
# SCPI: DIAGnostic:SIGNaling:LTE:CELL:LOGGing:MAC
driver.diagnostic.signaling.lte.cell.logging.mac.set(enable = False, log_type =␣
↪enums.LogType.DISable, payload = 1)
```

No command help available

param enable
No help available

param log_type
No help available

param payload
No help available

6.5.1.5.1.3 Pdcp

SCPI Command :

```
DIAGnostic:SIGNaling:LTE:CELL:LOGGing:PDCP
```

class PdcpCls

Pdcp commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class PdcpStruct

Response structure. Fields:

- Enable: bool: No parameter help available
- Log_Type: enums.LogType: No parameter help available
- Payload: int: No parameter help available

get() → PdcpStruct

```
# SCPI: DIAGnostic:SIGNaling:LTE:CELL:LOGGing:PDCP
value: PdcpStruct = driver.diagnostic.signaling.lte.cell.logging.pdcp.get()
```

No command help available

return
structure: for return value, see the help for PdcpStruct structure arguments.

set(*enable: bool*, *log_type: LogType*, *payload: int = None*) → None

```
# SCPI: DIAGnostic:SIGNaling:LTE:CELL:LOGGing:PDCP
driver.diagnostic.signaling.lte.cell.logging.pdcp.set(enable = False, log_type␣
↪= enums.LogType.DISable, payload = 1)
```

No command help available

param enable
No help available

param log_type
No help available

param payload
No help available

6.5.1.5.1.4 Rlc

SCPI Command :

```
DIAGnostic:SIGNaling:LTE:CELL:LOGGing:RLC
```

class RlcCls

Rlc commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class RlcStruct

Response structure. Fields:

- Enable: bool: No parameter help available
- Log_Type: enums.LogType: No parameter help available
- Payload: int: No parameter help available

get() → RlcStruct

```
# SCPI: DIAGnostic:SIGNaling:LTE:CELL:LOGGing:RLC
value: RlcStruct = driver.diagnostic.signaling.lte.cell.logging.rlc.get()
```

No command help available

return
structure: for return value, see the help for RlcStruct structure arguments.

set(*enable: bool*, *log_type: LogType*, *payload: int = None*) → None

```
# SCPI: DIAGnostic:SIGNaling:LTE:CELL:LOGGing:RLC
driver.diagnostic.signaling.lte.cell.logging.rlc.set(enable = False, log_type =␣
↪enums.LogType.DISable, payload = 1)
```

No command help available

param enable
No help available

param log_type
No help available

param payload
No help available

6.5.1.6 Nradio

class NradioCls

Nradio commands group definition. 3 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.diagnostic.signaling.nradio.clone()
```

Subgroups

6.5.1.6.1 Cell

class CellCls

Cell commands group definition. 3 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.diagnostic.signaling.nradio.cell.clone()
```

Subgroups

6.5.1.6.1.1 Logging

class LoggingCls

Logging commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.diagnostic.signaling.nradio.cell.logging.clone()
```

Subgroups

6.5.1.6.1.2 Mac

SCPI Command :

```
DIAGnostic:SIGNaling:NRADio:CELL:LOGGing:MAC
```

class MacCls

Mac commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class MacStruct

Response structure. Fields:

- Enable: bool: No parameter help available
- Log_Type: enums.LogType: No parameter help available
- Payload: int: No parameter help available

get() → MacStruct

```
# SCPI: DIAGnostic:SIGNaling:NRADio:CELL:LOGGing:MAC
value: MacStruct = driver.diagnostic.signaling.nradio.cell.logging.mac.get()
```

No command help available

return
structure: for return value, see the help for MacStruct structure arguments.

set(*enable: bool*, *log_type: LogType*, *payload: int = None*) → None

```
# SCPI: DIAGnostic:SIGNaling:NRADio:CELL:LOGGing:MAC
driver.diagnostic.signaling.nradio.cell.logging.mac.set(enable = False, log_
↪type = enums.LogType.DISable, payload = 1)
```

No command help available

param enable
No help available

param log_type
No help available

param payload
No help available

6.5.1.6.1.3 Pdcp

SCPI Command :

```
DIAGnostic:SIGNaling:NRADio:CELL:LOGGing:PDCP
```

class PdcpCls

Pdcp commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class PdcpStruct

Response structure. Fields:

- Enable: bool: No parameter help available
- Log_Type: enums.LogType: No parameter help available
- Payload: int: No parameter help available

get() → PdcpStruct

```
# SCPI: DIAGnostic:SIGNaling:NRADio:CELL:LOGGing:PDCP
value: PdcpStruct = driver.diagnostic.signaling.nradio.cell.logging.pdcp.get()
```

No command help available

return
structure: for return value, see the help for PdcpStruct structure arguments.

set(*enable: bool*, *log_type: LogType*, *payload: int = None*) → None

```
# SCPI: DIAGnostic:SIGNaling:NRADio:CELL:LOGGing:PDCP
driver.diagnostic.signaling.nradio.cell.logging.pdcp.set(enable = False, log_
↪type = enums.LogType.DISable, payload = 1)
```

No command help available

param enable
No help available

param log_type
No help available

param payload
No help available

6.5.1.6.1.4 Rlc

SCPI Command :

```
DIAGnostic:SIGNaling:NRADio:CELL:LOGGing:RLC
```

class RlcCls

Rlc commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class RlcStruct

Response structure. Fields:

- Enable: bool: No parameter help available
- Log_Type: enums.LogType: No parameter help available
- Payload: int: No parameter help available

get() → RlcStruct

```
# SCPI: DIAGnostic:SIGNaling:NRADio:CELL:LOGGing:RLC
value: RlcStruct = driver.diagnostic.signaling.nradio.cell.logging.rlc.get()
```

No command help available

return
structure: for return value, see the help for RlcStruct structure arguments.

set(*enable: bool*, *log_type: LogType*, *payload: int = None*) → None

```
# SCPI: DIAGnostic:SIGNaling:NRADio:CELL:LOGGing:RLC
driver.diagnostic.signaling.nradio.cell.logging.rlc.set(enable = False, log_
↪type = enums.LogType.DISable, payload = 1)
```

No command help available

param enable
No help available

param log_type
No help available

param payload
No help available

6.5.1.7 Register

class RegisterCls

Register commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.diagnostic.signaling.register.clone()
```

Subgroups

6.5.1.7.1 Existing

SCPI Command :

```
DIAGnostic:SIGNaling:REGister:EXISting
```

class ExistingCls

Existing commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*name_type: NameType = None*) → None

```
# SCPI: DIAGnostic:SIGNaling:REGister:EXISting
driver.diagnostic.signaling.register.existing.set(name_type = enums.NameType.
↪GUI)
```

No command help available

param name_type
No help available

6.5.1.8 Registration

SCPI Command :

```
DIAGnostic:SIGNaling:REGistration:RESet
```

class RegistrationCls

Registration commands group definition. 3 total commands, 2 Subgroups, 1 group commands

reset(*target: Target*) → None

```
# SCPI: DIAGnostic:SIGNaling:REGistration:RESet
driver.diagnostic.signaling.registration.reset(target = enums.Target.ALL)
```

No command help available

param target
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.diagnostic.signaling.registration.clone()
```

Subgroups

6.5.1.8.1 Add

SCPI Command :

```
DIAGnostic:SIGNaling:REGistration:ADD
```

class AddCls

Add commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*target: Target*, *name: str*, *resource: str*) → None

```
# SCPI: DIAGnostic:SIGNaling:REGistration:ADD
driver.diagnostic.signaling.registration.add.set(target = enums.Target.ALL,␣
↪name = 'abc', resource = 'abc')
```

No command help available

param target
No help available

param name
No help available

param resource
No help available

6.5.1.8.2 ListPy

SCPI Command :

```
DIAGnostic:SIGNaling:REGistration:LIST
```

class ListPyCls

ListPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*target: Target*) → List[str]

```
# SCPI: DIAGnostic:SIGNaling:REGistration:LIST
value: List[str] = driver.diagnostic.signaling.registration.listPy.get(target =␣
↪enums.Target.ALL)
```

No command help available

param target
No help available

return
item: No help available

6.5.1.9 Routing

SCPI Command :

```
DIAGnostic:SIGNaling:ROUTing
```

class RoutingCls

Routing commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class RoutingStruct

Response structure. Fields:

- Routing: enums.Routing: No parameter help available
- Cell_Signal: enums.DiagCellSignal: No parameter help available
- Baseband: enums.DiagBaseband: No parameter help available
- Tdd: enums.Tdd: No parameter help available

get() → RoutingStruct

```
# SCPI: DIAGnostic:SIGNaling:ROUTing
value: RoutingStruct = driver.diagnostic.signaling.routing.get()
```

No command help available

return
structure: for return value, see the help for RoutingStruct structure arguments.

set(*routing: Routing*, *cell_signal: DiagCellSignal = None*, *baseband: DiagBaseband = None*, *tdd: Tdd = None*) → None

```
# SCPI: DIAGnostic:SIGNaling:ROUTing
driver.diagnostic.signaling.routing.set(routing = enums.Routing.DUT, cell_
↪signal = enums.DiagCellSignal.COMBining, baseband = enums.DiagBaseband.
↪BBCombining, tdd = enums.Tdd.CP1)
```

No command help available

param routing
No help available

param cell_signal
No help available

param baseband
No help available

param tdd
No help available

6.5.1.10 Topology

class TopologyCls

Topology commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.diagnostic.signaling.topology.clone()
```

Subgroups

6.5.1.10.1 Subsriber

SCPI Command :

```
DIAGnostic:SIGNaling:TOPology:SUBSriber:CREation
```

class SubsriberCls

Subsriber commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_creation() → bool

```
# SCPI: DIAGnostic:SIGNaling:TOPology:SUBSriber:CREation
value: bool = driver.diagnostic.signaling.topology.subsriber.get_creation()
```

No command help available

return
skip: No help available

set_creation(*skip: bool*) → None

```
# SCPI: DIAGnostic:SIGNaling:TOPology:SUBSriber:CREation
driver.diagnostic.signaling.topology.subsriber.set_creation(skip = False)
```

No command help available

param skip
No help available

6.6 Init

class InitCls

Init commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.init.clone()
```

Subgroups

6.6.1 Signaling

class SignalingCls

Signaling commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.init.signaling.clone()
```

Subgroups

6.6.1.1 Measurement

class MeasurementCls

Measurement commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.init.signaling.measurement.clone()
```

Subgroups

6.6.1.1.1 CqiReporting

SCPI Command :

```
INIT:SIGNaling:MEASurement:CQIReporting
```

class CqiReportingCls

CqiReporting commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: INIT:SIGNaling:MEASurement:CQIReporting
driver.init.signaling.measurement.cqiReporting.set()
```

Starts the measurement. The measurement enters the 'RUN' state.

set_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: INIT:SIGNaling:MEASurement:CQIReporting
driver.init.signaling.measurement.cqiReporting.set_with_opc()
```

Starts the measurement. The measurement enters the 'RUN' state.

Same as set, but waits for the operation to complete before continuing further. Use the RsCMX_Signaling.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

6.7 Procedure

class ProcedureCls

Procedure commands group definition. 20 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.procedure.clone()
```

Subgroups

6.7.1 Signaling

class SignalingCls

Signaling commands group definition. 20 total commands, 7 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.procedure.signaling.clone()
```

Subgroups

6.7.1.1 ApMod

SCPI Command :

```
PROCedure:SIGNaling:APMod
```

class ApModCls

ApMod commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: PROCedure:SIGNaling:APMod
driver.procedure.signaling.apMod.set()
```

Applies all pending changes.

set_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: PROCedure:SIGNaling:APMod
driver.procedure.signaling.apMod.set_with_opc()
```

Applies all pending changes.

Same as set, but waits for the operation to complete before continuing further. Use the RsCMX_Signaling.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

6.7.1.2 Lte

class LteCls

Lte commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.procedure.signaling.lte.clone()
```

Subgroups

6.7.1.2.1 Cell

class CellCls

Cell commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.procedure.signaling.lte.cell.clone()
```

Subgroups

6.7.1.2.1.1 Power

class PowerCls

Power commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.procedure.signaling.lte.cell.power.clone()
```

Subgroups

6.7.1.2.1.2 Control

class ControlCls

Control commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.procedure.signaling.lte.cell.power.control.clone()
```

Subgroups

6.7.1.2.1.3 TpControl

class TpControlCls

TpControl commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.procedure.signaling.lte.cell.power.control.tpControl.clone()
```

Subgroups

6.7.1.2.1.4 Pattern

SCPI Command :

```
PROCedure:SIGNaling:LTE:CELL:POWer:CONTrol:TPControl:PATTern:EXECute
```

class PatternCls

Pattern commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set_execute(*cell_name: str*) → None

```
# SCPI: PROCedure:SIGNaling:LTE:CELL:POWer:CONTrol:TPControl:PATTern:EXECute
driver.procedure.signaling.lte.cell.power.control.tpControl.pattern.set_
↪execute(cell_name = 'abc')
```

Starts the execution of a user-defined TPC pattern.

param cell_name
No help available

6.7.1.3 Mobility

SCPI Command :

```
PROCedure:SIGNaling:MOBility:REDirection
```

class MobilityCls

Mobility commands group definition. 2 total commands, 1 Subgroups, 1 group commands

set_redirection(*target_cell: str*) → None

```
# SCPI: PROCedure:SIGNaling:MOBility:REDirection
driver.procedure.signaling.mobility.set_redirection(target_cell = 'abc')
```

Triggers a redirection from the current PCell to the <TargetCell>.

param target_cell
Name of the target cell of the redirection.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.procedure.signaling.mobility.clone()
```

Subgroups

6.7.1.3.1 Handover

SCPI Command :

```
PROCedure:SIGNaling:MOBility:HANDover
```

class HandoverCls

Handover commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*target_cell_mcg: str*, *target_cell_scg: TargetCellScg = None*) → None

```
# SCPI: PROCedure:SIGNaling:MOBility:HANDover
driver.procedure.signaling.mobility.handover.set(target_cell_mcg = 'abc',␣
↪target_cell_scg = enums.TargetCellScg.RELease)
```

Triggers a handover for the MCG, from the current PCell to the <TargetCellMcg>. And triggers SCG mobility, from the current PSCell to the <TargetCellScg>.

param target_cell_mcg
Name of the target cell for the MCG. If you want to keep the old PCell, enter KEEP.

param target_cell_scg
(enum or string) Configures SCG mobility. - string: If you have a PSCell and you want to change it, set the name of the target cell. - omit the parameter: If you have no PSCell or you want to keep it, skip the parameter. - RELease: If you want to drop the PSCell, set RELease.

6.7.1.4 Nradio

class NradioCls

Nradio commands group definition. 10 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.procedure.signaling.nradio.clone()
```

Subgroups

6.7.1.4.1 Cell

class CellCls

Cell commands group definition. 9 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.procedure.signaling.nradio.cell.clone()
```

Subgroups

6.7.1.4.1.1 Bwp<BwParts>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.procedure.signaling.nradio.cell.bwp.repcap_bwParts_get()
driver.procedure.signaling.nradio.cell.bwp.repcap_bwParts_set(repcap.BwParts.Nr1)
```

class BwpCls

Bwp commands group definition. 2 total commands, 1 Subgroups, 0 group commands Repeated Capability: BwParts, default value after init: BwParts.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.procedure.signaling.nradio.cell.bwp.clone()
```

Subgroups

6.7.1.4.1.2 Power

class PowerCls

Power commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.procedure.signaling.nradio.cell.bwp.power.clone()
```

Subgroups

6.7.1.4.1.3 Control

class ControlCls

Control commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.procedure.signaling.nradio.cell.bwp.power.control.clone()
```

Subgroups

6.7.1.4.1.4 TpControl

class TpControlCls

TpControl commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.procedure.signaling.nradio.cell.bwp.power.control.tpControl.clone()
```

Subgroups

6.7.1.4.1.5 Pattern

class PatternCls

Pattern commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.procedure.signaling.nradio.cell.bwp.power.control.tpControl.pattern.
↪clone()
```

Subgroups

6.7.1.4.1.6 Execute

SCPI Command :

```
PROCedure:SIGNaling:NRADio:CELL:BWP<bpwid>:POWer:CONTrol:TPControl:PATTern:EXECute
```

class ExecuteCls

Execute commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*cell_name: str*, *bwParts=BwParts.Default*) → None

```
# SCPI: PROCedure:SIGNaling:NRADio:CELL:BWP<bpwid>
↪:POWer:CONTrol:TPControl:PATTern:EXECute
driver.procedure.signaling.nradio.cell.bwp.power.control.tpControl.pattern.
↪execute.set(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Starts the execution of a user-defined TPC pattern, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.7.1.4.1.7 RpTolerance

class RpToleranceCls

RpTolerance commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.procedure.signaling.nradio.cell.bwp.power.control.tpControl.rpTolerance.
↪clone()
```

Subgroups

6.7.1.4.1.8 Execute

SCPI Command :

```
PROCedure:SIGNaling:NRADio:CELL:BWP<bpwid>:POWer:CONTrol:TPControl:RPTolerance:EXECute
```

class ExecuteCls

Execute commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*cell_name: str*, *bwParts=BwParts.Default*) → None

```
# SCPI: PROCedure:SIGNaling:NRADio:CELL:BWP<bpwid>
↪:POWer:CONTrol:TPControl:RPTolerance:EXECute
driver.procedure.signaling.nradio.cell.bwp.power.control.tpControl.rpTolerance.
↪execute.set(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Starts the execution of the TPC pattern for relative power tolerance tests, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.7.1.4.1.9 Cmatrix

SCPI Commands :

```
PROCedure:SIGNaling:NRADio:CELL:CMATrix:HADamard
PROCedure:SIGNaling:NRADio:CELL:CMATrix:TGPP
```

class CmatrixCls

Cmatrix commands group definition. 2 total commands, 0 Subgroups, 2 group commands

set_hadamard(*cell_name: str*) → None

```
# SCPI: PROCedure:SIGNaling:NRADio:CELL:CMATrix:HADamard
driver.procedure.signaling.nradio.cell.cmatrix.set_hadamard(cell_name = 'abc')
```

No command help available

param cell_name
No help available

set_tgpp(*cell_name: str*) → None

```
# SCPI: PROCedure:SIGNaling:NRADio:CELL:CMATrix:TGPP
driver.procedure.signaling.nradio.cell.cmatrix.set_tgpp(cell_name = 'abc')
```

No command help available

param cell_name
No help available

6.7.1.4.1.10 Power

class PowerCls

Power commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.procedure.signaling.nradio.cell.power.clone()
```

Subgroups

6.7.1.4.1.11 Control

class ControlCls

Control commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.procedure.signaling.nradio.cell.power.control.clone()
```

Subgroups

6.7.1.4.1.12 TpControl

class TpControlCls

TpControl commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.procedure.signaling.nradio.cell.power.control.tpControl.clone()
```

Subgroups

6.7.1.4.1.13 Pattern

SCPI Command :

```
PROCedure:SIGNaling:NRADio:CELL:POWer:CONTrol:TPControl:PATTern:EXECute
```

class PatternCls

Pattern commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set_execute(*cell_name: str*) → None

```
# SCPI: PROCedure:SIGNaling:NRADio:CELL:POWer:CONTrol:TPControl:PATTern:EXECute
driver.procedure.signaling.nradio.cell.power.control.tpControl.pattern.set_
↪execute(cell_name = 'abc')
```

Starts the execution of a user-defined TPC pattern, for the initial BWP.

param cell_name
No help available

6.7.1.4.1.14 RpTolerance

SCPI Command :

```
PROCedure:SIGNaling:NRADio:CELL:POWer:CONTrol:TPControl:RPTolerance:EXECute
```

class RpToleranceCls

RpTolerance commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set_execute(*cell_name: str*) → None

```
# SCPI:␣
↪PROCedure:SIGNaling:NRADio:CELL:POWer:CONTrol:TPControl:RPTolerance:EXECute
driver.procedure.signaling.nradio.cell.power.control.tpControl.rpTolerance.set_
↪execute(cell_name = 'abc')
```

Starts the execution of the TPC pattern for relative power tolerance tests, for the initial BWP.

param cell_name
No help available

6.7.1.4.1.15 VcCalib

SCPI Commands :

```
PROCedure:SIGNaling:NRADio:CELL:VCCalib:CALibrate
PROCedure:SIGNaling:NRADio:CELL:VCCalib:ISOLation
PROCedure:SIGNaling:NRADio:CELL:VCCalib:DEACtivate
```

class VcCalibCls

VcCalib commands group definition. 3 total commands, 0 Subgroups, 3 group commands

deactivate(*cell_name: str*) → None

```
# SCPI: PROCedure:SIGNaling:NRADio:CELL:VCCalib:DEACtivate
driver.procedure.signaling.nradio.cell.vcCalib.deactivate(cell_name = 'abc')
```

Discards the calibration matrix.

param cell_name
No help available

set_calibrate(*cell_name: str*) → None

```
# SCPI: PROCedure:SIGNaling:NRADio:CELL:VCCalib:CALibrate
driver.procedure.signaling.nradio.cell.vcCalib.set_calibrate(cell_name = 'abc')
```

Starts the virtual cable calibration.

param cell_name
No help available

set_isolation(*cell_name: str*) → None

```
# SCPI: PROCedure:SIGNaling:NRADio:CELL:VCCalib:ISOLation
driver.procedure.signaling.nradio.cell.vcCalib.set_isolation(cell_name = 'abc')
```

Updates the isolation quality information without changing the calibration matrix.

param cell_name
No help available

6.7.1.4.2 PdcchOrder

SCPI Command :

```
PROCedure:SIGNaling:NRADio:PDCChorder:ACTivate
```

class PdcchOrderCls

PdcchOrder commands group definition. 1 total commands, 0 Subgroups, 1 group commands

activate(*mode: ConfigMode = None*, *ra_preamble_idx: int = None*, *ssb_index: int = None*, *prach_mask_index: int = None*) → None

```
# SCPI: PROCedure:SIGNaling:NRADio:PDCChorder:ACTivate
driver.procedure.signaling.nradio.pdcchOrder.activate(mode = enums.ConfigMode.
↪AUTO, ra_preamble_idx = 1, ssb_index = 1, prach_mask_index = 1)
```

Triggers a PDCCH order for the primary NR cell (established connection needed) .

param mode
AUTO: Automatic configuration, ignore the remaining parameters. UDEFined: Configuration via the remaining parameters.

param ra_preamble_idx
Random access preamble index

param ssb_index
SS/PBCH index

param prach_mask_index
PRACH mask index

6.7.1.5 Nrdc

SCPI Commands :

```
PROCedure:SIGNaling:NRDC:ACTivate
PROCedure:SIGNaling:NRDC:DEACtivate
```

class NrdcCls

Nrdc commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ActivateStruct

Structure for setting input parameters. Contains optional setting parameters. Fields:

- Pscell: str: Optional setting parameter. Name of the second NR cell for which you want to activate NR-DC.
- Linked_Pdu_Id: int: Optional setting parameter. ID of the existing PDU session to which the QoS flow is added.
- Rlc_Mode: enums.RlcMode: Optional setting parameter. RLC mode ACK: acknowledged USACK: unacknowledged
- Data_Flow: enums.DataFlow: Optional setting parameter. Configures the user data flow via the master node (MN) and the secondary node (SN) . MCG: via MN, no traffic split MCGSplit: MCG split bearer, with traffic split in the MN SCG: via SN, no traffic split SCGSplit: SCG split bearer, with traffic split in the SN

- Traffic_Dist: float: Optional setting parameter. Configuration of a data flow with traffic split. A numeric value defines the percentage of the data to be transferred via the interface MN - UE. The remainder is transferred via the interface SN - UE. AUTO configures the traffic distribution automatically and dynamically, depending on the load in the MN path.
- Qi: enums.Qi: Optional setting parameter. 5G quality of service identifier (5QI) .
- Flow_Control: enums.FlowControl: Optional setting parameter. GUARanteed: GBR QoS flow NGUaranteed: non-GBR QoS flow
- Max_Dl_Bitrate: int: Optional setting parameter. Maximum flow bit rate (MFBR) for the DL.
- Max_Dl_Unit: enums.ItRateUnit: Optional setting parameter. Unit for 8_MaxDlBitrate. Kn, Mn, Gn, Tn, Pn = n kbit/s, Mbit/s, Gbit/s, Tbit/s, Pbit/s
- Max_Ul_Bitrate: int: Optional setting parameter. Maximum flow bit rate (MFBR) for the UL.
- Max_Ul_Unit: enums.ItRateUnit: Optional setting parameter. Unit for 10_MaxUlBitrate.
- Dl_Bitrate: int: Optional setting parameter. Guaranteed flow bit rate (GFBR) for the DL, only for GBR QoS flows.
- Dl_Unit: enums.ItRateUnit: Optional setting parameter. Unit for 12_DLBitrate, only for GBR QoS flows.
- Ul_Bitrate: int: Optional setting parameter. Guaranteed flow bit rate (GFBR) for the UL, only for GBR QoS flows.
- Ul_Unit: enums.ItRateUnit: Optional setting parameter. Unit for 14_ULBitrate, only for GBR QoS flows.
- Averaging_Window: int or bool: Optional setting parameter. Duration over which the bit rates GFBR and MFBR are calculated for GBR QoS flows. OFF omits the parameter in the QoS flow description.

activate(*structure: ActivateStruct*) → None

```
# SCPI: PROCedure:SIGNaling:NRDC:ACTivate
structure = driver.procedure.signaling.nrdc.ActivateStruct()
structure.Pscell: str = 'abc'
structure.Linked_Pdu_Id: int = 1
structure.Rlc_Mode: enums.RlcMode = enums.RlcMode.ACK
structure.Data_Flow: enums.DataFlow = enums.DataFlow.MCG
structure.Traffic_Dist: float = 1.0
structure.Qi: enums.Qi = enums.Qi.Q1
structure.Flow_Control: enums.FlowControl = enums.FlowControl.GUARanteed
structure.Max_Dl_Bitrate: int = 1
structure.Max_Dl_Unit: enums.ItRateUnit = enums.ItRateUnit.G1
structure.Max_Ul_Bitrate: int = 1
structure.Max_Ul_Unit: enums.ItRateUnit = enums.ItRateUnit.G1
structure.Dl_Bitrate: int = 1
structure.Dl_Unit: enums.ItRateUnit = enums.ItRateUnit.G1
structure.Ul_Bitrate: int = 1
structure.Ul_Unit: enums.ItRateUnit = enums.ItRateUnit.G1
structure.Averaging_Window: int or bool = 1
driver.procedure.signaling.nrdc.activate(structure)
```

Activates the NR-DC mode and creates a QoS flow.

param structure
for set value, see the help for ActivateStruct structure arguments.

deactivate(*pdu_session_id: int = None*, *qos_flow_id: int = None*) → None

```
# SCPI: PROCedure:SIGNaling:NRDC:DEACtivate
driver.procedure.signaling.nrdc.deactivate(pdu_session_id = 1, qos_flow_id = 1)
```

Deactivates the NR-DC mode and removes a QoS flow.

param pdu_session_id
ID of the PDU session from which the QoS flow is removed.

param qos_flow_id
ID of the QoS flow to be removed (QFI) .

6.7.1.6 Sms

SCPI Command :

```
PROCedure:SIGNaling:SMS
```

class SmsCls

Sms commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class SetStruct

Structure for setting input parameters. Contains optional setting parameters. Fields:

- Address: str: Address of the originator of the message
- Message: str: Message text
- Type_Py: enums.Type: Optional setting parameter. Coding group GDC: general data coding DCMC: data coding / message class
- Coding: enums.Coding: Optional setting parameter. Data coding, selecting the used character set GSM: GSM 7-bit default alphabet coding (ASCII) EIGHt: 8-bit binary data UCS2: UCS-2 16-bit coding (only for GDC, not for DCMC)
- Class_Py: enums.Class: Optional setting parameter. Message class 0 to 3, selecting to which component of the UE the message is delivered.
- Core_Network: enums.CoreNetwork: Optional setting parameter. Type of network delivering the message, EPS or 5G

set(*structure: SetStruct*) → None

```
# SCPI: PROCedure:SIGNaling:SMS
structure = driver.procedure.signaling.sms.SetStruct()
structure.Address: str = 'abc'
structure.Message: str = 'abc'
structure.Type_Py: enums.Type = enums.Type.DCMC
structure.Coding: enums.Coding = enums.Coding.EIGHt
structure.Class_Py: enums.Class = enums.Class.C0
structure.Core_Network: enums.CoreNetwork = enums.CoreNetwork.EPS
driver.procedure.signaling.sms.set(structure)
```

Sends a short message to the UE. For background information, see 3GPP TS 23.038.

param structure
for set value, see the help for SetStruct structure arguments.

6.7.1.7 Ue

class UeCls

Ue commands group definition. 3 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.procedure.signaling.ue.clone()
```

Subgroups

6.7.1.7.1 Rrc

SCPI Command :

```
PROCedure:SIGNaling:UE:RRC
```

class RrcCls

Rrc commands group definition. 3 total commands, 2 Subgroups, 1 group commands

set(*ue_id: str*, *action: Action*) → None

```
# SCPI: PROCedure:SIGNaling:UE:RRC
driver.procedure.signaling.ue.rrc.set(ue_id = 'abc', action = enums.Action.
↪CONNect)
```

Establishes or releases an RRC connection.

param ue_id
No help available

param action
DISConnect: release connection to idle CONNect: establish RRC connection

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.procedure.signaling.ue.rrc.clone()
```

Subgroups

6.7.1.7.1.1 Inactive

SCPI Command :

```
PROCedure:SIGNaling:UE:RRC:INACtive
```

class InactiveCls

Inactive commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*ue_id: str = None*, *rnau_timer: RnauTimer = None*, *paging_cycle: PagingCycle = None*) → None

```
# SCPI: PROCedure:SIGNaling:UE:RRC:INACtive
driver.procedure.signaling.ue.rrc.inactive.set(ue_id = 'abc', rnau_timer =␣
↪enums.RnauTimer.M10, paging_cycle = enums.PagingCycle.P128)
```

Suspends a 5G NR standalone RRC connection (RRC release with 'suspendConfig') , resulting in the RRC state Inactive.

param ue_id
No help available

param rnau_timer
RNAU timer triggering the periodic RAN-based notification area update, in minutes

param paging_cycle
UE-specific cycle for RAN-initiated paging, as number of radio frames

6.7.1.7.1.2 Resume

SCPI Command :

```
PROCedure:SIGNaling:UE:RRC:RESume
```

class ResumeCls

Resume commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: PROCedure:SIGNaling:UE:RRC:RESume
driver.procedure.signaling.ue.rrc.resume.set()
```

Resumes an inactive 5G NR standalone connection so that it becomes active again.

set_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: PROCedure:SIGNaling:UE:RRC:RESume
driver.procedure.signaling.ue.rrc.resume.set_with_opc()
```

Resumes an inactive 5G NR standalone connection so that it becomes active again.

Same as set, but waits for the operation to complete before continuing further. Use the RsCMX_Signaling.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

6.8 Remove

class RemoveCls

Remove commands group definition. 6 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.remove.clone()
```

Subgroups

6.8.1 Signaling

class SignalingCls

Signaling commands group definition. 6 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.remove.signaling.clone()
```

Subgroups

6.8.1.1 Lte

class LteCls

Lte commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.remove.signaling.lte.clone()
```

Subgroups

6.8.1.1.1 Ca

class CaCls

Ca commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.remove.signaling.lte.ca.clone()
```

Subgroups

6.8.1.1.1.1 Scell

SCPI Command :

```
REMove:SIGNaling:LTE:CA:SCELl
```

class ScellCls

Scell commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*cell_group_name: str*, *cell_name: str*) → None

```
# SCPI: REMove:SIGNaling:LTE:CA:SCELl
driver.remove.signaling.lte.ca.scell.set(cell_group_name = 'abc', cell_name =
↪'abc')
```

Removes an LTE or NR cell from a cell group.

param cell_group_name
No help available

param cell_name
No help available

6.8.1.1.2 Ncell

SCPI Command :

```
REMove:SIGNaling:LTE:NCELl
```

class NcellCls

Ncell commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*cell_name: str*, *ncell_name: str*) → None

```
# SCPI: REMove:SIGNaling:LTE:NCELl
driver.remove.signaling.lte.ncell.set(cell_name = 'abc', ncell_name = 'abc')
```

Removes a cell from the SIB neighbor cell list of an LTE or NR cell.

param cell_name
Name of the cell for which the neighbor is removed.

param ncell_name
Name of the neighbor cell.

6.8.1.2 Nradio

class NradioCls

Nradio commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.remove.signaling.nradio.clone()
```

Subgroups

6.8.1.2.1 Ca

class CaCls

Ca commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.remove.signaling.nradio.ca.clone()
```

Subgroups

6.8.1.2.1.1 Scell

SCPI Command :

```
REMove:SIGNaling:NRADio:CA:SCELl
```

class ScellCls

Scell commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*cell_group_name: str*, *cell_name: str*) → None

```
# SCPI: REMove:SIGNaling:NRADio:CA:SCELl
driver.remove.signaling.nradio.ca.scell.set(cell_group_name = 'abc', cell_name␣
↪= 'abc')
```

Removes an LTE or NR cell from a cell group.

param cell_group_name
No help available

param cell_name
No help available

6.8.1.2.2 Ncell

SCPI Command :

```
REMove:SIGNaling:NRADio:NCELl
```

class NcellCls

Ncell commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*cell_name: str*, *ncell_name: str*) → None

```
# SCPI: REMove:SIGNaling:NRADio:NCELl
driver.remove.signaling.nradio.ncell.set(cell_name = 'abc', ncell_name = 'abc')
```

Removes a cell from the SIB neighbor cell list of an LTE or NR cell.

param cell_name
Name of the cell for which the neighbor is removed.

param ncell_name
Name of the neighbor cell.

6.8.1.3 Topology

class TopologyCls

Topology commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.remove.signaling.topology.clone()
```

Subgroups

6.8.1.3.1 Eps

SCPI Command :

```
REMove:SIGNaling:TOPology:EPS
```

class EpsCls

Eps commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*name_ta_eps: str*, *name_cell: str*) → None

```
# SCPI: REMove:SIGNaling:TOPology:EPS
driver.remove.signaling.topology.eps.set(name_ta_eps = 'abc', name_cell = 'abc')
```

Removes the link between a cell and an EPS tracking area.

param name_ta_eps
No help available

param name_cell
No help available

6.8.1.3.2 Fgs

SCPI Command :

```
REMove:SIGNaling:TOPology:FGS
```

class FgsCls

Fgs commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*name_ta_5_g: str*, *name_cell: str*) → None

```
# SCPI: REMove:SIGNaling:TOPology:FGS
driver.remove.signaling.topology.fgs.set(name_ta_5_g = 'abc', name_cell = 'abc')
```

Removes the link between a cell and a 5GS tracking area.

param name_ta_5_g
No help available

param name_cell
No help available

6.9 Restart

class RestartCls

Restart commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.restart.clone()
```

Subgroups

6.9.1 Signaling

class SignalingCls

Signaling commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.restart.signaling.clone()
```

Subgroups

6.9.1.1 Topology

class TopologyCls

Topology commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.restart.signaling.topology.clone()
```

Subgroups

6.9.1.1.1 Cnetwork

SCPI Command :

```
RESTart:SIGNaling:TOPology:CNETwork
```

class CnetworkCls

Cnetwork commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: RESTart:SIGNaling:TOPology:CNETwork
driver.restart.signaling.topology.cnetwork.set()
```

Restarts the core network in live mode and resets the DUT states to Idle. Afterwards, you are back in live mode. The cells still exist and are switched off.

set_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: RESTart:SIGNaling:TOPology:CNETwork
driver.restart.signaling.topology.cnetwork.set_with_opc()
```

Restarts the core network in live mode and resets the DUT states to Idle. Afterwards, you are back in live mode. The cells still exist and are switched off.

Same as set, but waits for the operation to complete before continuing further. Use the RsCMX_Signaling.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

6.10 Sense

class SenseCls

Sense commands group definition. 44 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.clone()
```

Subgroups

6.10.1 Elog

SCPI Command :

```
SENSe:ELOG:ALL
```

class ElogCls

Elog commands group definition. 3 total commands, 2 Subgroups, 1 group commands

class AllStruct

Structure for reading output parameters. Fields:

- Severity: List[enums.Severity]: No parameter help available
- Timestamp: List[str]: No parameter help available
- Message: List[str]: No parameter help available

get_all() → AllStruct

```
# SCPI: SENSe:ELOG:ALL
value: AllStruct = driver.sense.elog.get_all()
```

No command help available

return
structure: for return value, see the help for AllStruct structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.elog.clone()
```

Subgroups

6.10.1.1 Last

SCPI Command :

```
SENSe:ELOG:LAST
```

class LastCls

Last commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Severity: List[enums.Severity]: No parameter help available
- Timestamp: List[str]: No parameter help available
- Message: List[str]: No parameter help available

get(*count: int = None*) → GetStruct

```
# SCPI: SENSe:ELOG:LAST
value: GetStruct = driver.sense.elog.last.get(count = 1)
```

No command help available

param count
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

6.10.1.2 Time

SCPI Command :

```
SENSe:ELOG:TIME
```

class TimeCls

Time commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Severity: List[enums.Severity]: No parameter help available
- Timestamp: List[str]: No parameter help available
- Message: List[str]: No parameter help available

get(*time_start: str*, *time_end: str*) → GetStruct

```
# SCPI: SENSe:ELOG:TIME
value: GetStruct = driver.sense.elog.time.get(time_start = 'abc', time_end =
↪'abc')
```

No command help available

param time_start
No help available

param time_end
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

6.10.2 Signaling

SCPI Command :

```
SENSe:SIGNaling:SMS
```

class SignalingCls

Signaling commands group definition. 41 total commands, 9 Subgroups, 1 group commands

class SmsStruct

Structure for reading output parameters. Fields:

- Core_Network: List[enums.CoreNetwork]: Type of network delivering the message, EPS or 5G
- Address: List[str]: Address of the originator of the message
- State: List[enums.StateTest]: States whether an error occurred.
- Message: List[str]: For successful transmission, the short message contents. For erroneous transmission, information about the error.

get_sms() → SmsStruct

```
# SCPI: SENSe:SIGNaling:SMS
value: SmsStruct = driver.sense.signaling.get_sms()
```

Queries information about the last received mobile-originated short message.

return
structure: for return value, see the help for SmsStruct structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.clone()
```

Subgroups

6.10.2.1 Awgn

class AwgnCls

Awgn commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.awgn.clone()
```

Subgroups

6.10.2.1.1 Advanced

class AdvancedCls

Advanced commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.awgn.advanced.clone()
```

Subgroups

6.10.2.1.1.1 Bandwidth

class BandwidthCls

Bandwidth commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.awgn.advanced.bandwidth.clone()
```

Subgroups

6.10.2.1.1.2 Noise

SCPI Command :

```
SENSe:SIGNaling:AWGN:ADVanced:BWIDth:NOISe
```

class NoiseCls

Noise commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → float

```
# SCPI: SENSe:SIGNaling:AWGN:ADVanced:BWIDth:NOISe
value: float = driver.sense.signaling.awgn.advanced.bandwidth.noise.get(cell_
↪name = 'abc')
```

Queries the noise bandwidth.

param cell_name
No help available

return
noise_bandwidth: No help available

6.10.2.2 Ccopy

class CcopyCls

Ccopy commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.ccopy.clone()
```

Subgroups

6.10.2.2.1 MccCopies

SCPI Command :

```
SENSe:SIGNaling:CCOPy:MCCCopies
```

class MccCopiesCls

MccCopies commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: SENSe:SIGNaling:CCOPy:MCCCopies
value: int = driver.sense.signaling.ccopy.mccCopies.get(cell_name = 'abc')
```

Queries the maximum number of contiguous cell copies for a specific source cell.

param cell_name
No help available

return
no_copies: Number of cells that fit into the same frequency band above the source cell (higher frequencies) .

6.10.2.3 Cell

class CellCls

Cell commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.cell.clone()
```

Subgroups

6.10.2.3.1 Instance

SCPI Command :

```
SENSe:SIGNaling:CELL:INSTance
```

class InstanceCls

Instance commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → str

```
# SCPI: SENSe:SIGNaling:CELL:INSTance
value: str = driver.sense.signaling.cell.instance.get(cell_name = 'abc')
```

Queries the default cell name corresponding to the current cell name. The command is useful to derive the name of trigger signals, see 'Trigger signals'.

param cell_name
Cell name that is used in most commands and GUI squares.

return
cell_instance: Default cell name. Used in names of trigger signals related to the cell.

6.10.2.4 Fading

class FadingCls

Fading commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.fading.clone()
```

Subgroups

6.10.2.4.1 Csamples

SCPI Command :

```
SENSe:SIGNaling:FADing:CSAMples
```

class CsamplesCls

Csamples commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → float

```
# SCPI: SENSe:SIGNaling:FADing:CSAMples
value: float = driver.sense.signaling.fading.csamples.get(cell_name = 'abc')
```

No command help available

param cell_name
No help available

return
clipped_samples: No help available

6.10.2.5 Lte

class LteCls

Lte commands group definition. 6 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.lte.clone()
```

Subgroups

6.10.2.5.1 Cell

class CellCls

Cell commands group definition. 6 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.lte.cell.clone()
```

Subgroups

6.10.2.5.1.1 BbgIndex

SCPI Command :

```
SENSe:SIGNaling:LTE:CELL:BBGindex
```

class BbgIndexCls

BbgIndex commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: SENSe:SIGNaling:LTE:CELL:BBGindex
value: int = driver.sense.signaling.lte.cell.bbgIndex.get(cell_name = 'abc')
```

Returns the number of the baseband group containing the cell. Cells in the same group use the same connectors.

param cell_name
No help available

return
bb_group_index: NAV means that the cell is not contained in a baseband group.

6.10.2.5.1.2 Harq

class HarqCls

Harq commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.lte.cell.harq.clone()
```

Subgroups

6.10.2.5.1.3 Downlink

class DownlinkCls

Downlink commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.lte.cell.harq.downlink.clone()
```

Subgroups

6.10.2.5.1.4 ReTx

class ReTxCls

ReTx commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.lte.cell.harq.downlink.reTx.clone()
```

Subgroups

6.10.2.5.1.5 Count

SCPI Command :

```
SENSe:SIGNaling:LTE:CELL:HARQ:DL:RETX:COUNt
```

class CountCls

Count commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: SENSe:SIGNaling:LTE:CELL:HARQ:DL:RETX:COUNt
value: int = driver.sense.signaling.lte.cell.harq.downlink.reTx.count.get(cell_
↪name = 'abc')
```

Query the number of entries in the retransmission configuration.

param cell_name
No help available

return
count: No help available

6.10.2.5.1.6 RvSequence

class RvSequenceCls

RvSequence commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.lte.cell.harq.downlink.rvSequence.clone()
```

Subgroups

6.10.2.5.1.7 Count

SCPI Command :

```
SENSe:SIGNaling:LTE:CELL:HARQ:DL:RVSequence:COUNt
```

class CountCls

Count commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: SENSe:SIGNaling:LTE:CELL:HARQ:DL:RVSequence:COUNt
value: int = driver.sense.signaling.lte.cell.harq.downlink.rvSequence.count.
↪get(cell_name = 'abc')
```

Query the length (number of entries) of the RV sequences.

param cell_name
No help available

return
count: No help available

6.10.2.5.1.8 Power

class PowerCls

Power commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.lte.cell.power.clone()
```

Subgroups

6.10.2.5.1.9 Downlink

class DownlinkCls

Downlink commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.lte.cell.power.downlink.clone()
```

Subgroups

6.10.2.5.1.10 Maximum

SCPI Command :

```
SENSe:SIGNaling:LTE:CELL:POWer:DL:MAXimum
```

class MaximumCls

Maximum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → float

```
# SCPI: SENSe:SIGNaling:LTE:CELL:POWer:DL:MAXimum
value: float = driver.sense.signaling.lte.cell.power.downlink.maximum.get(cell_
↪name = 'abc')
```

No command help available

param cell_name
No help available

return
max_cell_power: No help available

6.10.2.5.1.11 UeScheduling

class UeSchedulingCls

UeScheduling commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.lte.cell.ueScheduling.clone()
```

Subgroups

6.10.2.5.1.12 Dynamic

class DynamicCls

Dynamic commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.lte.cell.ueScheduling.dynamic.clone()
```

Subgroups

6.10.2.5.1.13 Downlink

class DownlinkCls

Downlink commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.lte.cell.ueScheduling.dynamic.downlink.clone()
```

Subgroups

6.10.2.5.1.14 TypePy

SCPI Command :

```
SENSe:SIGNaling:LTE:CELL:UESCheduling:DYNamic:DL:TYPE
```

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Follow_Cqi: enums.FollowCqi: DISabled: No follow CQI. WB: Follow WB CQI. MSB: Follow maximum SB CQI. UEPSubband: Follow UE-preferred SB CQI. UEBSubband: Follow the best SB CQI.

- Follow_Pmi: enums.FollowPmi: DISabled: No follow PMI. WB: Follow WB PMI. WBEXplicit: Follow WB PMI and PMI sent to the UE via DCI. SB: Follow SB PMI.
- Follow_Ri: enums.FollowRi: DISabled: No follow RI. ENABled: New RI applied after the current retransmission cycle. RETX: New RI applied immediately, also for retransmissions.

get(*cell_name: str*) → GetStruct

```
# SCPI: SENSe:SIGNaling:LTE:CELL:UEScheduling:DYNamic:DL:TYPE
value: GetStruct = driver.sense.signaling.lte.cell.ueScheduling.dynamic.
↪downlink.typePy.get(cell_name = 'abc')
```

Queries which follow modes are active for the DL.

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

6.10.2.5.1.15 TypePy

SCPI Command :

```
SENSe:SIGNaling:LTE:CELL:UEScheduling:TYPE
```

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Type_Dl: enums.TypeDlUl: For downlink
- Type_Ul: enums.TypeDlUl: For uplink

get(*cell_name: str*) → GetStruct

```
# SCPI: SENSe:SIGNaling:LTE:CELL:UEScheduling:TYPE
value: GetStruct = driver.sense.signaling.lte.cell.ueScheduling.typePy.get(cell_
↪name = 'abc')
```

Queries whether the downlink and uplink scheduling settings correspond to an RMC defined by 3GPP (RMC) or not (UDEFined) .

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

6.10.2.6 Nradio

class NradioCls

Nradio commands group definition. 26 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.nradio.clone()
```

Subgroups

6.10.2.6.1 Ca

class CaCls

Ca commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.nradio.ca.clone()
```

Subgroups

6.10.2.6.1.1 Dormancy

class DormancyCls

Dormancy commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.nradio.ca.dormancy.clone()
```

Subgroups

6.10.2.6.1.2 State

SCPI Command :

```
SENSe:SIGNaling:NRADio:CA:DORMancy:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_group_name: str*) → bool

```
# SCPI: SENSe:SIGNaling:NRADio:CA:DORMancy:STATe
value: bool = driver.sense.signaling.nradio.ca.dormancy.state.get(cell_group_
↪name = 'abc')
```

Queries the dormancy state of a cell group.

param cell_group_name
No help available

return
dormant_state: Dormant (ON) or non-dormant (OFF) .

6.10.2.6.2 Cell

class CellCls

Cell commands group definition. 25 total commands, 11 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.nradio.cell.clone()
```

Subgroups

6.10.2.6.2.1 Alayout

class AlayoutCls

Alayout commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.nradio.cell.alayout.clone()
```

Subgroups

6.10.2.6.2.2 Ptype

SCPI Command :

```
SENSe:SIGNaling:NRADio:CELL:ALAYout:PTYPe
```

class PtypeCls

Ptype commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → PannelType

```
# SCPI: SENSe:SIGNaling:NRADio:CELL:ALAYout:PTYPe
value: enums.PannelType = driver.sense.signaling.nradio.cell.alayout.ptype.
↪get(cell_name = 'abc')
```

No command help available

param cell_name
No help available

return
pannel_type: No help available

6.10.2.6.2.3 BbgIndex

SCPI Command :

```
SENSe:SIGNaling:NRADio:CELL:BBGindex
```

class BbgIndexCls

BbgIndex commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: SENSe:SIGNaling:NRADio:CELL:BBGindex
value: int = driver.sense.signaling.nradio.cell.bbgIndex.get(cell_name = 'abc')
```

Returns the number of the baseband group containing the cell. Cells in the same group use the same connectors.

param cell_name
No help available

return
bb_group_index: NAV means that the cell is not contained in a baseband group.

6.10.2.6.2.4 Beams

class BeamsCls

Beams commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.nradio.cell.beams.clone()
```

Subgroups

6.10.2.6.2.5 ActiveBeam

SCPI Command :

```
SENSe:SIGNaling:NRADio:CELL:BEAMs:ACTivebeam
```

class ActiveBeamCls

ActiveBeam commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Association: enums.Association: SSB beam or NZP CSI-RS beam
- Index: int: Beam index

get(*cell_name: str*) → GetStruct

```
# SCPI: SENSe:SIGNaling:NRADio:CELL:BEAMs:ACTivebeam
value: GetStruct = driver.sense.signaling.nradio.cell.beams.activeBeam.get(cell_
↪name = 'abc')
```

Queries information about the active beam.

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

6.10.2.6.2.6 Bwp<BwParts>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.sense.signaling.nradio.cell.bwp.repcap_bwParts_get()
driver.sense.signaling.nradio.cell.bwp.repcap_bwParts_set(repcap.BwParts.Nr1)
```

class BwpCls

Bwp commands group definition. 10 total commands, 6 Subgroups, 0 group commands Repeated Capability: BwParts, default value after init: BwParts.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.nradio.cell.bwp.clone()
```

Subgroups

6.10.2.6.2.7 CqiReporting

class CqiReportingCls

CqiReporting commands group definition. 3 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.nradio.cell.bwp.cqiReporting.clone()
```

Subgroups

6.10.2.6.2.8 Report

class ReportCls

Report commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.nradio.cell.bwp.cqiReporting.report.clone()
```

Subgroups

6.10.2.6.2.9 Periodicity

SCPI Command :

```
SENSe:SIGNaling:NRADio:CELL:BWP<bwp_id>:CQIReporting:REPort:PERiodicity
```

class PeriodicityCls

Periodicity commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → PeriodicityCqiReport

```
# SCPI: SENSe:SIGNaling:NRADio:CELL:BWP<bwp_id>:CQIReporting:REPort:PERiodicity
value: enums.PeriodicityCqiReport = driver.sense.signaling.nradio.cell.bwp.
↪cqiReporting.report.periodicity.get(cell_name = 'abc', bwParts = repcap.
↪BwParts.Default)
```

No command help available

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
periodicity: No help available

6.10.2.6.2.10 Quantity

SCPI Command :

```
SENSe:SIGNaling:NRADio:CELL:BWP<bwp_id>:CQIReporting:REPort:QUANtity
```

class QuantityCls

Quantity commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → Quantity

```
# SCPI: SENSe:SIGNaling:NRADio:CELL:BWP<bwp_id>:CQIReporting:REPort:QUANtity
value: enums.Quantity = driver.sense.signaling.nradio.cell.bwp.cqiReporting.
↪report.quantity.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

No command help available

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
quantity: No help available

6.10.2.6.2.11 Resource

class ResourceCls

Resource commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.nradio.cell.bwp.cqiReporting.resource.clone()
```

Subgroups

6.10.2.6.2.12 Periodicity

SCPI Command :

```
SENSe:SIGNaling:NRADio:CELL:BWP<bwp_id>:CQIReporting:RESource:PERiodicity
```

class PeriodicityCls

Periodicity commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → PeriodicityRsrc

```
# SCPI: SENSe:SIGNaling:NRADio:CELL:BWP<bwp_id>
↪:CQIReporting:RESource:PERiodicity
value: enums.PeriodicityRsrc = driver.sense.signaling.nradio.cell.bwp.
↪cqiReporting.resource.periodicity.get(cell_name = 'abc', bwParts = repcap.
↪BwParts.Default)
```

No command help available

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
periodicity: No help available

6.10.2.6.2.13 Harq

class HarqCls

Harq commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.nradio.cell.bwp.harq.clone()
```

Subgroups

6.10.2.6.2.14 Downlink

class DownlinkCls

Downlink commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.nradio.cell.bwp.harq.downlink.clone()
```

Subgroups

6.10.2.6.2.15 User

class UserCls

User commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.nradio.cell.bwp.harq.downlink.user.clone()
```

Subgroups

6.10.2.6.2.16 Retransm

class RetransmCls

Retransm commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.nradio.cell.bwp.harq.downlink.user.retransm.clone()
```

Subgroups

6.10.2.6.2.17 Count

SCPI Command :

```
SENSe:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:DL:USER:RETRansm:COUNt
```

class CountCls

Count commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → int

```
# SCPI: SENSe:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:DL:USER:RETRansm:COUNt
value: int = driver.sense.signaling.nradio.cell.bwp.harq.downlink.user.retransm.
↪count.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Queries the number of DL retransmissions for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
retransmissions: No help available

6.10.2.6.2.18 Uplink

class UplinkCls

Uplink commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.nradio.cell.bwp.harq.uplink.clone()
```

Subgroups

6.10.2.6.2.19 User

class UserCls

User commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.nradio.cell.bwp.harq.uplink.user.clone()
```

Subgroups

6.10.2.6.2.20 Retransm

class RetransmCls

Retransm commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.nradio.cell.bwp.harq.uplink.user.retransm.clone()
```

Subgroups

6.10.2.6.2.21 Count

SCPI Command :

```
SENSe:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:UL:USER:RETRansm:COUNt
```

class CountCls

Count commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → int

```
# SCPI: SENSe:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:UL:USER:RETRansm:COUNt
value: int = driver.sense.signaling.nradio.cell.bwp.harq.uplink.user.retransm.
↪count.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Queries the number of UL retransmissions for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
retransmissions: No help available

6.10.2.6.2.22 Id

SCPI Command :

```
SENSe:SIGNaling:NRADio:CELL:BWP<bwpid>:ID
```

class IdCls

Id commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → int

```
# SCPI: SENSe:SIGNaling:NRADio:CELL:BWP<bwpid>:ID
value: int = driver.sense.signaling.nradio.cell.bwp.id.get(cell_name = 'abc',␣
↪bwParts = repcap.BwParts.Default)
```

No command help available

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
bwp_id: No help available

6.10.2.6.2.23 Power

class PowerCls

Power commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.nradio.cell.bwp.power.clone()
```

Subgroups

6.10.2.6.2.24 Control

class ControlCls

Control commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.nradio.cell.bwp.power.control.clone()
```

Subgroups

6.10.2.6.2.25 TpControl

class TpControlCls

TpControl commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.nradio.cell.bwp.power.control.tpControl.clone()
```

Subgroups

6.10.2.6.2.26 RpTolerance

SCPI Command :

```
SENSe:SIGNaling:NRADio:CELL:BWP<bpwid>:POWer:CONTrol:TPControl:RPTolerance
```

class RpToleranceCls

RpTolerance commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Start_Power: float: Initial power level to which the UE is commanded before executing the actual TPC pattern.
- Length: int: Number of active UL slots in the TPC pattern (after the start power) .
- Rb_Change_Pos: int: Position of the RB allocation change within the TPC pattern (number of UL subframes) .

get(*cell_name: str*, *bwParts=BwParts.Default*) → GetStruct

```
# SCPI: SENSe:SIGNaling:NRADio:CELL:BWP<bpwid>
↪:POWer:CONTrol:TPControl:RPTolerance
value: GetStruct = driver.sense.signaling.nradio.cell.bwp.power.control.
↪tpControl.rpTolerance.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Queries information about the TPC pattern configured for relative power tolerance tests for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
structure: for return value, see the help for GetStruct structure arguments.

6.10.2.6.2.27 Pucch

class PucchCls

Pucch commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.nradio.cell.bwp.pucch.clone()
```

Subgroups

6.10.2.6.2.28 Nsymbols

SCPI Command :

```
SENSe:SIGNaling:NRADio:CELL:BWP<bpwid>:PUCCh:NSYMbols
```

class NsymbolsCls

Nsymbols commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → int

```
# SCPI: SENSe:SIGNaling:NRADio:CELL:BWP<bpwid>:PUCCh:NSYMbols
value: int = driver.sense.signaling.nradio.cell.bwp.pucch.nsymbols.get(cell_
↪name = 'abc', bwParts = repcap.BwParts.Default)
```

Queries the number of allocated OFDM symbols resulting from the PUCCH format, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
no_symbols: No help available

6.10.2.6.2.29 SsIndex

SCPI Command :

```
SENSe:SIGNaling:NRADio:CELL:BWP<bpwid>:PUCCh:SSINdex
```

class SsIndexCls

SsIndex commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*, *bwParts=BwParts.Default*) → int

```
# SCPI: SENSe:SIGNaling:NRADio:CELL:BWP<bpwid>:PUCCh:SSINdex
value: int = driver.sense.signaling.nradio.cell.bwp.pucch.ssIndex.get(cell_name␣
↪= 'abc', bwParts = repcap.BwParts.Default)
```

Queries the index of the first allocated symbol resulting from the PUCCH format, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
starting_sym_idx: No help available

6.10.2.6.2.30 UeScheduling

class UeSchedulingCls

UeScheduling commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.nradio.cell.bwp.ueScheduling.clone()
```

Subgroups

6.10.2.6.2.31 Dynamic

class DynamicCls

Dynamic commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.nradio.cell.bwp.ueScheduling.dynamic.clone()
```

Subgroups

6.10.2.6.2.32 TypePy

SCPI Command :

```
SENSe:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:DYNamic:TYPE
```

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Follow_Cqi: enums.FollowType: No parameter help available
- Follow_Pmi: enums.FollowType: No parameter help available
- Follow_Ri: enums.FollowType: No parameter help available
- Follow_Bo: enums.FollowType: No parameter help available

get(*cell_name: str*, *bwParts=BwParts.Default*) → GetStruct

```
# SCPI: SENSe:SIGNaling:NRADio:CELL:BWP<bwp_id>:UESCheduling:DYNamic:TYPE
value: GetStruct = driver.sense.signaling.nradio.cell.bwp.ueScheduling.dynamic.
↪typePy.get(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Queries which follow modes are active, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
structure: for return value, see the help for GetStruct structure arguments.

6.10.2.6.2.33 CqiReporting

class CqiReportingCls

CqiReporting commands group definition. 3 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.nradio.cell.cqiReporting.clone()
```

Subgroups

6.10.2.6.2.34 Report

class ReportCls

Report commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.nradio.cell.cqiReporting.report.clone()
```

Subgroups

6.10.2.6.2.35 Periodicity

SCPI Command :

```
SENSe:SIGNaling:NRADio:CELL:CQIReporting:REPort:PERiodicity
```

class PeriodicityCls

Periodicity commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → PeriodicityCqiReport

```
# SCPI: SENSe:SIGNaling:NRADio:CELL:CQIReporting:REPort:PERiodicity
value: enums.PeriodicityCqiReport = driver.sense.signaling.nradio.cell.
↪cqiReporting.report.periodicity.get(cell_name = 'abc')
```

Queries the periodicity of CSI reports, configured indirectly via [CONFigure:]SIGNaling:NRADio:CELL:CQIReporting:PERiodicity, for the initial BWP.

param cell_name
No help available

return
periodicity: Periodicity in slots

6.10.2.6.2.36 Quantity

SCPI Command :

```
SENSe:SIGNaling:NRADio:CELL:CQIReporting:REPort:QUANtity
```

class QuantityCls

Quantity commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → Quantity

```
# SCPI: SENSe:SIGNaling:NRADio:CELL:CQIReporting:REPort:QUANtity
value: enums.Quantity = driver.sense.signaling.nradio.cell.cqiReporting.report.
↪quantity.get(cell_name = 'abc')
```

No command help available

param cell_name
No help available

return
quantity: No help available

6.10.2.6.2.37 Resource

class ResourceCls

Resource commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.nradio.cell.cqiReporting.resource.clone()
```

Subgroups

6.10.2.6.2.38 Periodicity

SCPI Command :

```
SENSe:SIGNaling:NRADio:CELL:CQIReporting:RESource:PERiodicity
```

class PeriodicityCls

Periodicity commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → PeriodicityRsrc

```
# SCPI: SENSe:SIGNaling:NRADio:CELL:CQIReporting:RESource:PERiodicity
value: enums.PeriodicityRsrc = driver.sense.signaling.nradio.cell.cqiReporting.
↪resource.periodicity.get(cell_name = 'abc')
```

Queries the periodicity of CSI-RS, configured indirectly via [CONFigure:]SIGNaling:NRADio:CELL:CQIReporting:PERiodicity, for the initial BWP.

param cell_name
No help available

return
periodicity: Periodicity in slots

6.10.2.6.2.39 Harq

class HarqCls

Harq commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.nradio.cell.harq.clone()
```

Subgroups

6.10.2.6.2.40 Downlink

class DownlinkCls

Downlink commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.nradio.cell.harq.downlink.clone()
```

Subgroups

6.10.2.6.2.41 User

class UserCls

User commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.nradio.cell.harq.downlink.user.clone()
```

Subgroups

6.10.2.6.2.42 Retransm

class RetransmCls

Retransm commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.nradio.cell.harq.downlink.user.retransm.clone()
```

Subgroups

6.10.2.6.2.43 Count

SCPI Command :

```
SENSe:SIGNaling:NRADio:CELL:HARQ:DL:USER:RETRansm:COUNt
```

class CountCls

Count commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: SENSe:SIGNaling:NRADio:CELL:HARQ:DL:USER:RETRansm:COUNt
value: int = driver.sense.signaling.nradio.cell.harq.downlink.user.retransm.
↪count.get(cell_name = 'abc')
```

Queries the number of DL retransmissions for the initial BWP.

param cell_name
No help available

return
retransmissions: No help available

6.10.2.6.2.44 Uplink

class UplinkCls

Uplink commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.nradio.cell.harq.uplink.clone()
```

Subgroups

6.10.2.6.2.45 User

class UserCls

User commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.nradio.cell.harq.uplink.user.clone()
```

Subgroups

6.10.2.6.2.46 Retransm

class RetransmCls

Retransm commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.nradio.cell.harq.uplink.user.retransm.clone()
```

Subgroups

6.10.2.6.2.47 Count

SCPI Command :

```
SENSe:SIGNaling:NRADio:CELL:HARQ:UL:USER:RETRansm:COUNt
```

class CountCls

Count commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: SENSe:SIGNaling:NRADio:CELL:HARQ:UL:USER:RETRansm:COUNt
value: int = driver.sense.signaling.nradio.cell.harq.uplink.user.retransm.count.
↪get(cell_name = 'abc')
```

Queries the number of UL retransmissions for the initial BWP.

param cell_name
No help available

return
retransmissions: No help available

6.10.2.6.2.48 Power

class PowerCls

Power commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.nradio.cell.power.clone()
```

Subgroups

6.10.2.6.2.49 Control

class ControlCls

Control commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.nradio.cell.power.control.clone()
```

Subgroups

6.10.2.6.2.50 TpControl

class TpControlCls

TpControl commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.nradio.cell.power.control.tpControl.clone()
```

Subgroups

6.10.2.6.2.51 RpTolerance

SCPI Command :

```
SENSe:SIGNaling:NRADio:CELL:POWer:CONTrol:TPControl:RPTolerance
```

class RpToleranceCls

RpTolerance commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Start_Power: float: Initial power level to which the UE is commanded before executing the actual TPC pattern.
- Length: int: Number of active UL slots in the TPC pattern (after the start power) .
- Rb_Change_Pos: int: Position of the RB allocation change within the TPC pattern (number of UL subframes) .

get(*cell_name: str*) → GetStruct

```
# SCPI: SENSe:SIGNaling:NRADio:CELL:POWer:CONTrol:TPControl:RPTolerance
value: GetStruct = driver.sense.signaling.nradio.cell.power.control.tpControl.
↪rpTolerance.get(cell_name = 'abc')
```

Queries information about the TPC pattern configured for relative power tolerance tests for the initial BWP.

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

6.10.2.6.2.52 Pucch

class PucchCls

Pucch commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.nradio.cell.pucch.clone()
```

Subgroups

6.10.2.6.2.53 Nsymbols

SCPI Command :

```
SENSe:SIGNaling:NRADio:CELL:PUCCh:NSYMbols
```

class NsymbolsCls

Nsymbols commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: SENSe:SIGNaling:NRADio:CELL:PUCCh:NSYMbols
value: int = driver.sense.signaling.nradio.cell.pucch.nsymbols.get(cell_name =
↪'abc')
```

Queries the number of allocated OFDM symbols resulting from the PUCCH format, for the initial BWP.

param cell_name
No help available

return
no_symbols: No help available

6.10.2.6.2.54 SsIndex

SCPI Command :

```
SENSe:SIGNaling:NRADio:CELL:PUCCh:SSINdex
```

class SsIndexCls

SsIndex commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → int

```
# SCPI: SENSe:SIGNaling:NRADio:CELL:PUCCh:SSINdex
value: int = driver.sense.signaling.nradio.cell.pucch.ssIndex.get(cell_name =
↪'abc')
```

Queries the index of the first allocated symbol resulting from the PUCCH format, for the initial BWP.

param cell_name
No help available

return
starting_sym_idx: No help available

6.10.2.6.2.55 RfSettings

class RfSettingsCls

RfSettings commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.nradio.cell.rfSettings.clone()
```

Subgroups

6.10.2.6.2.56 Cfrequency

SCPI Command :

```
SENSe:SIGNaling:NRADio:CELL:RFSettings:CFRequency
```

class CfrequencyCls

Cfrequency commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Dl_Frequency: float: DL carrier center frequency
- Ul_Frequency: float: UL carrier center frequency

get(*cell_name: str*) → GetStruct

```
# SCPI: SENSe:SIGNaling:NRADio:CELL:RFSettings:CFRequency
value: GetStruct = driver.sense.signaling.nradio.cell.rfSettings.cfrequency.
↪get(cell_name = 'abc')
```

Queries the carrier center frequencies.

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

6.10.2.6.2.57 Ssb

class SsbCls

Ssb commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.nradio.cell.ssb.clone()
```

Subgroups

6.10.2.6.2.58 Beam

class BeamCls

Beam commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.nradio.cell.ssb.beam.clone()
```

Subgroups

6.10.2.6.2.59 Pbitmap

SCPI Command :

```
SENSe:SIGNaling:NRADio:CELL:SSB:BEAM:PBITmap
```

class PbitmapCls

Pbitmap commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Position_In_Burst: str: Bitmap, 0 = not transmitted, 1 = transmitted
- Active_Beam_Index: int: SSB index of active beam (leftmost bit in the bitmap has index 0) .

get(*cell_name: str*) → GetStruct

```
# SCPI: SENSe:SIGNaling:NRADio:CELL:SSB:BEAM:PBITmap
value: GetStruct = driver.sense.signaling.nradio.cell.ssb.beam.pbitmap.get(cell_
↪name = 'abc')
```

Queries the position bitmap, showing the time domain positions of the transmitted SS-blocks.

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

6.10.2.6.2.60 UeScheduling

class UeSchedulingCls

UeScheduling commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.nradio.cell.ueScheduling.clone()
```

Subgroups

6.10.2.6.2.61 Dynamic

class DynamicCls

Dynamic commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.nradio.cell.ueScheduling.dynamic.clone()
```

Subgroups

6.10.2.6.2.62 TypePy

SCPI Command :

```
SENSe:SIGNaling:NRADio:CELL:UESCheduling:DYNamic:TYPE
```

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Follow_Cqi: enums.FollowType: No parameter help available
- Follow_Pmi: enums.FollowType: No parameter help available
- Follow_Ri: enums.FollowType: No parameter help available
- Follow_Bo: enums.FollowType: No parameter help available

get(*cell_name: str*) → GetStruct

```
# SCPI: SENSe:SIGNaling:NRADio:CELL:UESCheduling:DYNamic:TYPE
value: GetStruct = driver.sense.signaling.nradio.cell.ueScheduling.dynamic.
↪typePy.get(cell_name = 'abc')
```

Queries which follow modes are active, for the initial BWP.

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

6.10.2.6.2.63 TypePy

SCPI Command :

```
SENSe:SIGNaling:NRADio:CELL:UESCheduling:TYPE
```

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Type_Dl: enums.TypeDlUl: For downlink
- Type_Ul: enums.TypeDlUl: For uplink

get(*cell_name: str*) → GetStruct

```
# SCPI: SENSe:SIGNaling:NRADio:CELL:UESCheduling:TYPE
value: GetStruct = driver.sense.signaling.nradio.cell.ueScheduling.typePy.
↪get(cell_name = 'abc')
```

Queries whether the downlink and uplink scheduling settings correspond to an RMC defined by 3GPP (RMC) or not (UDEFined) .

param cell_name
No help available

return
structure: for return value, see the help for GetStruct structure arguments.

6.10.2.7 Tmode

SCPI Command :

```
SENSe:SIGNaling:TMODe:SSReport
```

class TmodeCls

Tmode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class SsReportStruct

Structure for reading output parameters. Fields:

- Measured_Ssb_Id: float: ID of the SSB for which the RSRP values are reported.
- Branch_0: float: RSRP value reported by the UE for its receiver branch 0.
- Branch_1: float: RSRP value reported by the UE for its receiver branch 1.

get_ss_report() → SsReportStruct

```
# SCPI: SENSe:SIGNaling:TMODe:SSReport
value: SsReportStruct = driver.sense.signaling.tmode.get_ss_report()
```

Queries SS-RSRPB report contents received from the UE.

return
structure: for return value, see the help for SsReportStruct structure arguments.

6.10.2.8 Topology

class TopologyCls

Topology commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.topology.clone()
```

Subgroups

6.10.2.8.1 Eps

class EpsCls

Eps commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.topology.eps.clone()
```

Subgroups

6.10.2.8.1.1 Ue

SCPI Commands :

```
SENSe:SIGNaling:TOPology:EPS:UE:IMSI
SENSe:SIGNaling:TOPology:EPS:UE:IMEI
```

class UeCls

Ue commands group definition. 2 total commands, 0 Subgroups, 2 group commands

get_imei() → str

```
# SCPI: SENSe:SIGNaling:TOPology:EPS:UE:IMEI
value: str = driver.sense.signaling.topology.eps.ue.get_imei()
```

No command help available

return
imei: No help available

get_imsi() → str

```
# SCPI: SENSe:SIGNaling:TOPology:EPS:UE:IMSI
value: str = driver.sense.signaling.topology.eps.ue.get_imsi()
```

Queries the IMSI of the UE.

return
imsi: No help available

6.10.2.9 Ue

class UeCls

Ue commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.signaling.ue.clone()
```

Subgroups

6.10.2.9.1 Connection

SCPI Command :

```
SENSe:SIGNaling:UE:CONNection:UEPower
```

class ConnectionCls

Connection commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class UePowerStruct

Structure for reading output parameters. Fields:

- Status: enums.PowerStatus: No parameter help available
- Power_Level: List[float]: No parameter help available

get_ue_power() → UePowerStruct

```
# SCPI: SENSe:SIGNaling:UE:CONNection:UEPower
value: UePowerStruct = driver.sense.signaling.ue.connection.get_ue_power()
```

No command help available

return
structure: for return value, see the help for UePowerStruct structure arguments.

6.11 Signaling

class SignalingCls

Signaling commands group definition. 80 total commands, 12 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.clone()
```

Subgroups

6.11.1 Awgn

class AwgnCls

Awgn commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.awgn.clone()
```

Subgroups

6.11.1.1 Advanced

SCPI Command :

```
DELete:SIGNaling:AWGN:ADVanced
```

class AdvancedCls

Advanced commands group definition. 1 total commands, 0 Subgroups, 1 group commands

delete(*cell_name: str*) → None

```
# SCPI: DELete:SIGNaling:AWGN:ADVanced
driver.signaling.awgn.advanced.delete(cell_name = 'abc')
```

No command help available

param cell_name
No help available

6.11.2 Eps

class EpsCls

Eps commands group definition. 3 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.eps.clone()
```

Subgroups

6.11.2.1 UeCapability

class UeCapabilityCls

UeCapability commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.eps.ueCapability.clone()
```

Subgroups

6.11.2.1.1 Eutra

class EutraCls

Eutra commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.eps.ueCapability.eutra.clone()
```

Subgroups

6.11.2.1.1.1 Bands

SCPI Command :

```
DELete:SIGNaling:EPS:UECapability:EUTRa:BANDs
```

class BandsCls

Bands commands group definition. 1 total commands, 0 Subgroups, 1 group commands

delete(*fbi: List[int]*) → None

```
# SCPI: DELete:SIGNaling:EPS:UECapability:EUTRa:BANDs
driver.signaling.eps.ueCapability.eutra.bands.delete(fbi = [1, 2, 3])
```

Deletes entries from the list of requested frequency bands for the container type 'UE-EUTRA-Capability', for EPS tracking areas.

param fbi
Comma-separated list of LTE frequency band indicators

6.11.2.1.2 Mrdc

class MrdcCls

Mrdc commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.eps.ueCapability.mrdc.clone()
```

Subgroups

6.11.2.1.2.1 Bands

SCPI Command :

```
DELete:SIGNaling:EPS:UECapability:MRDC:BANDs
```

class BandsCls

Bands commands group definition. 1 total commands, 0 Subgroups, 1 group commands

delete(*rat: List[CellType]*, *fbi: List[int]*) → None

```
# SCPI: DELete:SIGNaling:EPS:UECapability:MRDC:BANDs
driver.signaling.eps.ueCapability.mrdc.bands.delete(rat = [CellType.LTE,␣
↪CellType.NR], fbi = [1, 2, 3])
```

Deletes entries from the list of requested frequency bands for the container type 'UE-MRDC-Capability', for EPS tracking areas. The bands are defined as pairs of values: {<Rat>, <Fbi>}1, {<Rat>, <Fbi>}2, ...

param rat
Type of the band: LTE band or NR band.

param fbi
Frequency band indicator

6.11.2.1.3 Nradio

class NradioCls

Nradio commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.eps.ueCapability.nradio.clone()
```

Subgroups

6.11.2.1.3.1 Bands

SCPI Command :

```
DELete:SIGNaling:EPS:UECapability:NRADio:BANDs
```

class BandsCls

Bands commands group definition. 1 total commands, 0 Subgroups, 1 group commands

delete(*fbi: List[int]*) → None

```
# SCPI: DELete:SIGNaling:EPS:UECapability:NRADio:BANDs
driver.signaling.eps.ueCapability.nradio.bands.delete(fbi = [1, 2, 3])
```

Deletes entries from the list of requested frequency bands for the container type 'UE-NR-Capability', for EPS tracking areas.

param fbi
Comma-separated list of NR frequency band indicators

6.11.3 Fading

SCPI Command :

```
DELete:SIGNaling:FADing
```

class FadingCls

Fading commands group definition. 1 total commands, 0 Subgroups, 1 group commands

delete(*cell_name: str*) → None

```
# SCPI: DELete:SIGNaling:FADing
driver.signaling.fading.delete(cell_name = 'abc')
```

Forbids any fading and releases the resources reserved for fading. Send this command before switching to live mode.

param cell_name
No help available

6.11.4 Fgs

class FgsCls

Fgs commands group definition. 3 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.fgs.clone()
```

Subgroups

6.11.4.1 UeCapability

class UeCapabilityCls

UeCapability commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.fgs.ueCapability.clone()
```

Subgroups

6.11.4.1.1 Eutra

class EutraCls

Eutra commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.fgs.ueCapability.eutra.clone()
```

Subgroups

6.11.4.1.1.1 Bands

SCPI Command :

```
DELete:SIGNaling:FGS:UECapability:EUTRa:BANDs
```

class BandsCls

Bands commands group definition. 1 total commands, 0 Subgroups, 1 group commands

delete(*fbi: List[int]*) → None

```
# SCPI: DELete:SIGNaling:FGS:UECapability:EUTRa:BANDs
driver.signaling.fgs.ueCapability.eutra.bands.delete(fbi = [1, 2, 3])
```

Deletes entries from the list of requested frequency bands for the container type 'UE-EUTRA-Capability', for 5GS tracking areas.

param fbi
Comma-separated list of LTE frequency band indicators

6.11.4.1.2 Mrdc

class MrdcCls

Mrdc commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.fgs.ueCapability.mrdc.clone()
```

Subgroups

6.11.4.1.2.1 Bands

SCPI Command :

```
DELete:SIGNaling:FGS:UECapability:MRDC:BANDs
```

class BandsCls

Bands commands group definition. 1 total commands, 0 Subgroups, 1 group commands

delete(*rat: List[CellType]*, *fbi: List[int]*) → None

```
# SCPI: DELete:SIGNaling:FGS:UECapability:MRDC:BANDs
driver.signaling.fgs.ueCapability.mrdc.bands.delete(rat = [CellType.LTE,␣
↪CellType.NR], fbi = [1, 2, 3])
```

Deletes entries from the list of requested frequency bands for the container type 'UE-MRDC-Capability', for 5GS tracking areas. The bands are defined as pairs of values: {<Rat>, <Fbi>}1, {<Rat>, <Fbi>}2, ...

param rat
Type of the band: LTE band or NR band.

param fbi
Frequency band indicator

6.11.4.1.3 Nradio

class NradioCls

Nradio commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.fgs.ueCapability.nradio.clone()
```

Subgroups

6.11.4.1.3.1 Bands

SCPI Command :

```
DELete:SIGNaling:FGS:UECapability:NRADio:BANDs
```

class BandsCls

Bands commands group definition. 1 total commands, 0 Subgroups, 1 group commands

delete(*fbi: List[int]*) → None

```
# SCPI: DELete:SIGNaling:FGS:UECapability:NRADio:BANDs
driver.signaling.fgs.ueCapability.nradio.bands.delete(fbi = [1, 2, 3])
```

Deletes entries from the list of requested frequency bands for the container type 'UE-NR-Capability', for 5GS tracking areas.

param fbi
Comma-separated list of NR frequency band indicators

6.11.5 Log

class LogCls

Log commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.log.clone()
```

Subgroups

6.11.5.1 File

class FileCls

File commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.log.file.clone()
```

Subgroups

6.11.5.1.1 Latest

SCPI Command :

```
FETCh:SIGNaling:LOG:FILE:LATest
```

class LatestCls

Latest commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[str]

```
# SCPI: FETCh:SIGNaling:LOG:FILE:LATest
value: List[str] = driver.signaling.log.file.latest.fetch()
```

No command help available

return
log_files: No help available

6.11.5.1.2 State

SCPI Command :

```
FETCh:SIGNaling:LOG:FILE:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → LogFileState

```
# SCPI: FETCh:SIGNaling:LOG:FILE:STATe
value: enums.LogFileState = driver.signaling.log.file.state.fetch()
```

No command help available

return
state: No help available

6.11.6 Lte

class LteCls

Lte commands group definition. 5 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.lte.clone()
```

Subgroups

6.11.6.1 Cell

SCPI Command :

```
DELete:SIGNaling:LTE:CELL
```

class CellCls

Cell commands group definition. 4 total commands, 2 Subgroups, 1 group commands

delete(*cell_name: str*) → None

```
# SCPI: DELete:SIGNaling:LTE:CELL
driver.signaling.lte.cell.delete(cell_name = 'abc')
```

Deletes an LTE or NR cell.

param cell_name
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.lte.cell.clone()
```

Subgroups

6.11.6.1.1 Harq

class HarqCls

Harq commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.lte.cell.harq.clone()
```

Subgroups

6.11.6.1.1.1 Downlink

class DownlinkCls

Downlink commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.lte.cell.harq.downlink.clone()
```

Subgroups

6.11.6.1.1.2 ReTx

SCPI Command :

```
DELete:SIGNaling:LTE:CELL:HARQ:DL:RETX
```

class ReTxCls

ReTx commands group definition. 1 total commands, 0 Subgroups, 1 group commands

delete(*cell_name: str*, *index: int*, *count: int = None*) → None

```
# SCPI: DELete:SIGNaling:LTE:CELL:HARQ:DL:RETX
driver.signaling.lte.cell.harq.downlink.reTx.delete(cell_name = 'abc', index =␣
↪1, count = 1)
```

Removes a block of entries from the retransmission configuration.

param cell_name
No help available

param index
Index of the first entry to be removed (lowest index is 0) .

param count
Number of entries to be removed (default is 1) .

6.11.6.1.1.3 RvSequence

SCPI Command :

```
DELete:SIGNaling:LTE:CELL:HARQ:DL:RVSequence
```

class RvSequenceCls

RvSequence commands group definition. 1 total commands, 0 Subgroups, 1 group commands

delete(*cell_name: str*, *index: int*, *count: int = None*) → None

```
# SCPI: DELete:SIGNaling:LTE:CELL:HARQ:DL:RVSequence
driver.signaling.lte.cell.harq.downlink.rvSequence.delete(cell_name = 'abc',␣
↪index = 1, count = 1)
```

Removes a block of entries from the RV sequences.

param cell_name
No help available

param index
Index of the first entry to be removed (lowest index is 0) .

param count
Number of entries to be removed (default is 1) .

6.11.6.1.2 Power

class PowerCls

Power commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.lte.cell.power.clone()
```

Subgroups

6.11.6.1.2.1 Control

class ControlCls

Control commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.lte.cell.power.control.clone()
```

Subgroups

6.11.6.1.2.2 State

SCPI Command :

```
FETCh:SIGNaling:LTE:CELL:POWer:CONTrol:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*cell_name: str*) → StatePwrControl

```
# SCPI: FETCh:SIGNaling:LTE:CELL:POWer:CONTrol:STATe
value: enums.StatePwrControl = driver.signaling.lte.cell.power.control.state.
↪fetch(cell_name = 'abc')
```

Queries whether a TPC power control procedure is running. For example, whether commanding the UE to maximum power is still ongoing or already complete.

param cell_name
No help available

return
state: Ready or running

6.11.6.2 Cgroup

SCPI Command :

```
DELete:SIGNaling:LTE:CGRoup
```

class CgroupCls

Cgroup commands group definition. 1 total commands, 0 Subgroups, 1 group commands

delete(*cell_group_name: str*) → None

```
# SCPI: DELete:SIGNaling:LTE:CGRoup
driver.signaling.lte.cgroup.delete(cell_group_name = 'abc')
```

Deletes an LTE or NR cell group.

param cell_group_name
No help available

6.11.7 Measurement

class MeasurementCls

Measurement commands group definition. 29 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.measurement.clone()
```

Subgroups

6.11.7.1 Bler

SCPI Commands :

```
ABORt:SIGNaling:MEASurement:BLER
STOP:SIGNaling:MEASurement:BLER
INITiate:SIGNaling:MEASurement:BLER
```

class BlerCls

Bler commands group definition. 21 total commands, 8 Subgroups, 3 group commands

abort() → None

```
# SCPI: ABORt:SIGNaling:MEASurement:BLER
driver.signaling.measurement.bler.abort()
```

Stops the measurement. The measurement enters the 'RDY' state.

abort_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: ABORt:SIGNaling:MEASurement:BLER
driver.signaling.measurement.bler.abort_with_opc()
```

Stops the measurement. The measurement enters the 'RDY' state.

Same as abort, but waits for the operation to complete before continuing further. Use the RsCMX_Signaling.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

initiate() → None

```
# SCPI: INITiate:SIGNaling:MEASurement:BLER
driver.signaling.measurement.bler.initiate()
```

Starts the measurement. The measurement enters the 'RUN' state.

initiate_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: INITiate:SIGNaling:MEASurement:BLER
driver.signaling.measurement.bler.initiate_with_opc()
```

Starts the measurement. The measurement enters the 'RUN' state.

Same as initiate, but waits for the operation to complete before continuing further. Use the RsCMX_Signaling.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

stop() → None

```
# SCPI: STOP:SIGNaling:MEASurement:BLER
driver.signaling.measurement.bler.stop()
```

No command help available

stop_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: STOP:SIGNaling:MEASurement:BLER
driver.signaling.measurement.bler.stop_with_opc()
```

No command help available

Same as stop, but waits for the operation to complete before continuing further. Use the RsCMX_Signaling.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.measurement.bler.clone()
```

Subgroups

6.11.7.1.1 Absolute

SCPI Command :

```
FETCh:SIGNaling:MEASurement:BLER:ABSolute
```

class AbsoluteCls

Absolute commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability indicator'

- Cell_Name: List[str]: Name of the cell providing the measured connection
- Ack: List[int]: Number of received acknowledgments
- Nack: List[int]: Number of received negative acknowledgments
- Dtx: List[int]: Number of missing answers (no ACK, no NACK)
- Throughput_Avg: List[int]: Average throughput in bit/s

fetch() → FetchStruct

```
# SCPI: FETCh:SIGNaling:MEASurement:BLER:ABSolute
value: FetchStruct = driver.signaling.measurement.bler.absolute.fetch()
```

Returns the absolute DL results of the BLER measurement. There is one set of results {...} per cell: <Reliability>, {<CellName>, <ACK>, <NACK>, <DTX>, <ThroughputAvg>}, {...}, ...

return
structure: for return value, see the help for FetchStruct structure arguments.

6.11.7.1.2 Confidence

SCPI Command :

```
FETCh:SIGNaling:MEASurement:BLER:CONFidence
```

class ConfidenceCls

Confidence commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability indicator'
- Cell_Name: List[str]: Name of the cell providing the measured connection
- State: List[enums.BlerState]: PENDing: measurement still running, no verdict yet PASS, FAIL: verdict of the measurement

fetch() → FetchStruct

```
# SCPI: FETCh:SIGNaling:MEASurement:BLER:CONFidence
value: FetchStruct = driver.signaling.measurement.bler.confidence.fetch()
```

Returns the results of a confidence BLER measurement. There is one set of results {...} per cell: <Reliability>, {<CellName>, <State>}, {...}, ...

return
structure: for return value, see the help for FetchStruct structure arguments.

6.11.7.1.3 Cword<Cword>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.signaling.measurement.bler.cword.repcap_cword_get()
driver.signaling.measurement.bler.cword.repcap_cword_set(repcap.Cword.Nr1)
```

class CwordCls

Cword commands group definition. 3 total commands, 3 Subgroups, 0 group commands Repeated Capability: Cword, default value after init: Cword.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.measurement.bler.cword.clone()
```

Subgroups

6.11.7.1.3.1 Absolute

SCPI Command :

```
FETCh:SIGNaling:MEASurement:BLER:CWORd<no>:ABSolute
```

class AbsoluteCls

Absolute commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability indicator'
- Cell_Name: List[str]: Name of the cell providing the measured connection
- Ack: List[int]: Number of received acknowledgments
- Nack: List[int]: Number of received negative acknowledgments
- Dtx: List[int]: Number of missing answers (no ACK, no NACK)
- Throughput_Avg: List[int]: Average throughput in bit/s

fetch(*cword=Cword.Default*) → FetchStruct

```
# SCPI: FETCh:SIGNaling:MEASurement:BLER:CWORd<no>:ABSolute
value: FetchStruct = driver.signaling.measurement.bler.cword.absolute.
↪fetch(cword = repcap.Cword.Default)
```

Returns the absolute DL results of the BLER measurement, for code word <no>. There is one set of results {...} per cell: <Reliability>, {<CellName>, <ACK>, <NACK>, <DTX>, <ThroughputAvg>}, {...}, ...

param cword
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cword')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.11.7.1.3.2 Relative

SCPI Command :

```
FETCh:SIGNaling:MEASurement:BLER:CWORd<no>:RELative
```

class RelativeCls

Relative commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability indicator'
- Cell_Name: List[str]: Name of the cell providing the measured connection
- Ack: List[float]: Number of received acknowledgments as percentage
- Nack: List[float]: Number of received negative acknowledgments as percentage
- Dtx: List[float]: Number of missing answers (no ACK, no NACK) as percentage
- Bler: List[float]: Block error ratio as percentage
- Throughput_Avg: List[float]: Average throughput as percentage of scheduled throughput

fetch(*algorithm: Algorithm = None*, *cword=Cword.Default*) → FetchStruct

```
# SCPI: FETCh:SIGNaling:MEASurement:BLER:CWORd<no>:RELative
value: FetchStruct = driver.signaling.measurement.bler.cword.relative.
↪fetch(algorithm = enums.Algorithm.ERC1, cword = repcap.Cword.Default)
```

Returns the relative DL results of the BLER measurement, for code word <no>. There is one set of results {...} per cell: <Reliability>, {<CellName>, <ACK>, <NACK>, <DTX>, <BLER>, <ThroughputAvg>}, {...}, ...

param algorithm
Selects the formula for calculation of the BLER from the number of ACK, NACK and DTX. ERC1 (Default) : BLER = (NACK + DTX) / (ACK + NACK + DTX) ERC2: BLER = DTX / (ACK + NACK + DTX) ERC3: BLER = NACK / (ACK + NACK + DTX) ERC4: BLER = NACK / (ACK + NACK)

param cword
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cword')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.11.7.1.3.3 Throughput

SCPI Command :

```
FETCh:SIGNaling:MEASurement:BLER:CWORd<no>:THRoughput
```

class ThroughputCls

Throughput commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability indicator'
- Cell_Name: List[str]: Name of the cell providing the measured connection
- Rel_Avg: List[float]: Average throughput as percentage of scheduled throughput
- Abs_Ack: List[int]: Average throughput in bit/s
- Abs_Scheduled: List[int]: Scheduled throughput in bit/s

fetch(*cword=Cword.Default*) → FetchStruct

```
# SCPI: FETCh:SIGNaling:MEASurement:BLER:CWORd<no>:THRoughput
value: FetchStruct = driver.signaling.measurement.bler.cword.throughput.
↪fetch(cword = repcap.Cword.Default)
```

Returns the DL throughput results of the BLER measurement, for code word <no>. There is one set of results {... } per cell: <Reliability>, {<CellName>, <RelAvg>, <AbsAck>, <AbsScheduled>}, {...}, ...

param cword
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cword')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.11.7.1.4 Overall

class OverallCls

Overall commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.measurement.bler.overall.clone()
```

Subgroups

6.11.7.1.4.1 Absolute

SCPI Command :

```
FETCh:SIGNaling:MEASurement:BLER:OVERall:ABSolute
```

class AbsoluteCls

Absolute commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability indicator'
- Ack: int: Number of received acknowledgments
- Nack: int: Number of received negative acknowledgments
- Dtx: int: Number of missing answers (no ACK, no NACK)
- Throughput_Avg: int: Average throughput in bit/s

fetch() → FetchStruct

```
# SCPI: FETCh:SIGNaling:MEASurement:BLER:OVERall:ABSolute
value: FetchStruct = driver.signaling.measurement.bler.overall.absolute.fetch()
```

Returns the overall absolute DL results of the BLER measurement.

return
structure: for return value, see the help for FetchStruct structure arguments.

6.11.7.1.4.2 Confidence

SCPI Command :

```
FETCh:SIGNaling:MEASurement:BLER:OVERall:CONFidence
```

class ConfidenceCls

Confidence commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → BlerState

```
# SCPI: FETCh:SIGNaling:MEASurement:BLER:OVERall:CONFidence
value: enums.BlerState = driver.signaling.measurement.bler.overall.confidence.
↪fetch()
```

Returns the overall results of a confidence BLER measurement.

Suppressed linked return values: reliability

return
state: PENDing: measurement still running, no verdict yet PASS, FAIL: verdict of the measurement

6.11.7.1.4.3 Relative

SCPI Command :

```
FETCh:SIGNaling:MEASurement:BLER:OVERall:RELative
```

class RelativeCls

Relative commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability indicator'
- Ack: float: Number of received acknowledgments as percentage
- Nack: float: Number of received negative acknowledgments as percentage
- Dtx: float: Number of missing answers (no ACK, no NACK) as percentage
- Bler: float: Block error ratio as percentage
- Throughput_Avg: float: Average throughput as percentage of scheduled throughput

fetch(*algorithm: Algorithm = None*) → FetchStruct

```
# SCPI: FETCh:SIGNaling:MEASurement:BLER:OVERall:RELative
value: FetchStruct = driver.signaling.measurement.bler.overall.relative.
↪fetch(algorithm = enums.Algorithm.ERC1)
```

Returns the overall relative DL results of the BLER measurement.

param algorithm
Selects the formula for calculation of the BLER from the number of ACK, NACK and DTX. ERC1 (Default) : BLER = (NACK + DTX) / (ACK + NACK + DTX) ERC2: BLER = DTX / (ACK + NACK + DTX) ERC3: BLER = NACK / (ACK + NACK + DTX) ERC4: BLER = NACK / (ACK + NACK)

return
structure: for return value, see the help for FetchStruct structure arguments.

6.11.7.1.4.4 Throughput

SCPI Command :

```
FETCh:SIGNaling:MEASurement:BLER:OVERall:THRoughput
```

class ThroughputCls

Throughput commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability indicator'
- Rel_Avg: float: Average throughput as percentage of scheduled throughput
- Abs_Ack: int: Average throughput in bit/s

- Abs_Scheduled: int: Scheduled throughput in bit/s

fetch() → FetchStruct

```
# SCPI: FETCh:SIGNaling:MEASurement:BLER:OVERall:THRoughput
value: FetchStruct = driver.signaling.measurement.bler.overall.throughput.
↪fetch()
```

Returns the overall DL throughput results of the BLER measurement.

return
structure: for return value, see the help for FetchStruct structure arguments.

6.11.7.1.5 Relative

SCPI Command :

```
FETCh:SIGNaling:MEASurement:BLER:RELative
```

class RelativeCls

Relative commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability indicator'
- Cell_Name: List[str]: Name of the cell providing the measured connection
- Ack: List[float]: Number of received acknowledgments as percentage
- Nack: List[float]: Number of received negative acknowledgments as percentage
- Dtx: List[float]: Number of missing answers (no ACK, no NACK) as percentage
- Bler: List[float]: Block error ratio as percentage
- Throughput_Avg: List[float]: Average throughput as percentage of scheduled throughput

fetch(*algorithm: Algorithm = None*) → FetchStruct

```
# SCPI: FETCh:SIGNaling:MEASurement:BLER:RELative
value: FetchStruct = driver.signaling.measurement.bler.relative.fetch(algorithm␣
↪= enums.Algorithm.ERC1)
```

Returns the relative DL results of the BLER measurement. There is one set of results {...} per cell: <Reliability>, {<CellName>, <ACK>, <NACK>, <DTX>, <BLER>, <ThroughputAvg>}, {...}, ...

param algorithm
Selects the formula for calculation of the BLER from the number of ACK, NACK and DTX. ERC1 (Default) : BLER = (NACK + DTX) / (ACK + NACK + DTX) ERC2: BLER = DTX / (ACK + NACK + DTX) ERC3: BLER = NACK / (ACK + NACK + DTX) ERC4: BLER = NACK / (ACK + NACK)

return
structure: for return value, see the help for FetchStruct structure arguments.

6.11.7.1.6 State

SCPI Command :

```
FETCh:SIGNaling:MEASurement:BLER:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- State: enums.State: OFF: Measurement off, no results. RDY: Measurement finished, valid results can be available. RUN: Measurement running.
- Cell_Name: str: No parameter help available
- Cell_Type: enums.CellType: No parameter help available

fetch(*info: Info = None*) → FetchStruct

```
# SCPI: FETCh:SIGNaling:MEASurement:BLER:STATe
value: FetchStruct = driver.signaling.measurement.bler.state.fetch(info = enums.
↪Info.ALL)
```

Queries the measurement state.

param info
No help available

return
structure: for return value, see the help for FetchStruct structure arguments.

6.11.7.1.7 Throughput

SCPI Command :

```
FETCh:SIGNaling:MEASurement:BLER:THRoughput
```

class ThroughputCls

Throughput commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability indicator'
- Cell_Name: List[str]: Name of the cell providing the measured connection
- Rel_Avg: List[float]: Average throughput as percentage of scheduled throughput
- Abs_Ack: List[int]: Average throughput in bit/s
- Abs_Scheduled: List[int]: Scheduled throughput in bit/s

fetch() → FetchStruct

```
# SCPI: FETCh:SIGNaling:MEASurement:BLER:THRoughput
value: FetchStruct = driver.signaling.measurement.bler.throughput.fetch()
```

Returns the DL throughput results of the BLER measurement. There is one set of results {...} per cell: <Reliability>, {<CellName>, <RelAvg>, <AbsAck>, <AbsScheduled>}, {...}, ...

return
structure: for return value, see the help for FetchStruct structure arguments.

6.11.7.1.8 Uplink

class UplinkCls

Uplink commands group definition. 6 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.measurement.bler.uplink.clone()
```

Subgroups

6.11.7.1.8.1 Absolute

SCPI Command :

```
FETCh:SIGNaling:MEASurement:BLER:UL:ABSolute
```

class AbsoluteCls

Absolute commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability indicator'
- Cell_Name: List[str]: Name of the cell providing the measured connection
- Crc_Failed: List[int]: Number of failed CRC
- Crc_Passed: List[int]: Number of passed CRC
- Dtx: List[int]: Discontinuous transmissions

fetch() → FetchStruct

```
# SCPI: FETCh:SIGNaling:MEASurement:BLER:UL:ABSolute
value: FetchStruct = driver.signaling.measurement.bler.uplink.absolute.fetch()
```

Returns the absolute UL results of the BLER measurement. There is one set of results {...} per cell: <Reliability>, {<CellName>, <CRCFailed>, <CRCPassed>, <DTX>}, {...}, ...

return
structure: for return value, see the help for FetchStruct structure arguments.

6.11.7.1.8.2 Overall

class OverallCls

Overall commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.measurement.bler.uplink.overall.clone()
```

Subgroups

6.11.7.1.8.3 Absolute

SCPI Command :

```
FETCh:SIGNaling:MEASurement:BLER:UL:OVERall:ABSolute
```

class AbsoluteCls

Absolute commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability indicator'
- Crc_Passed: int: Number of passed CRC
- Crc_Failed: int: Number of failed CRC
- Dtx: int: Discontinuous transmissions

fetch() → FetchStruct

```
# SCPI: FETCh:SIGNaling:MEASurement:BLER:UL:OVERall:ABSolute
value: FetchStruct = driver.signaling.measurement.bler.uplink.overall.absolute.
↪fetch()
```

Returns the overall absolute UL results of the BLER measurement.

return
structure: for return value, see the help for FetchStruct structure arguments.

6.11.7.1.8.4 Relative

SCPI Command :

```
FETCh:SIGNaling:MEASurement:BLER:UL:OVERall:RELative
```

class RelativeCls

Relative commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability indicator'
- Crc_Passed: float: Number of passed CRC as a percentage
- Crc_Failed: float: Number of failed CRC as a percentage
- Dtx: float: Discontinuous transmissions as a percentage
- Bler: float: Block error ratio as a percentage

fetch() → FetchStruct

```
# SCPI: FETCh:SIGNaling:MEASurement:BLER:UL:OVERall:RELative
value: FetchStruct = driver.signaling.measurement.bler.uplink.overall.relative.
↪fetch()
```

Returns the overall relative UL results of the BLER measurement.

return
structure: for return value, see the help for FetchStruct structure arguments.

6.11.7.1.8.5 Throughput

SCPI Command :

```
FETCh:SIGNaling:MEASurement:BLER:UL:OVERall:THRoughput
```

class ThroughputCls

Throughput commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability indicator'
- Rel_Avg: float: Throughput as percentage of scheduled throughput
- Abs_Crc_Passed: int: Throughput in bit/s
- Abs_Scheduled: int: Scheduled throughput in bit/s

fetch() → FetchStruct

```
# SCPI: FETCh:SIGNaling:MEASurement:BLER:UL:OVERall:THRoughput
value: FetchStruct = driver.signaling.measurement.bler.uplink.overall.
↪throughput.fetch()
```

Returns the overall UL throughput results of the BLER measurement.

return
structure: for return value, see the help for FetchStruct structure arguments.

6.11.7.1.8.6 Relative

SCPI Command :

```
FETCh:SIGNaling:MEASurement:BLER:UL:RELative
```

class RelativeCls

Relative commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability indicator'
- Cell_Name: List[str]: Name of the cell providing the measured connection
- Crc_Failed: List[int]: Number of failed CRC as a percentage
- Crc_Passed: List[int]: Number of passed CRC as a percentage
- Dtx: List[int]: Discontinuous transmissions as a percentage
- Bler: List[float]: Block error ratio as a percentage

fetch() → FetchStruct

```
# SCPI: FETCh:SIGNaling:MEASurement:BLER:UL:RELative
value: FetchStruct = driver.signaling.measurement.bler.uplink.relative.fetch()
```

Returns the relative UL results of the BLER measurement. There is one set of results {...} per cell: <Reliability>, {<CellName>, <CRCFailed>, <CRCPassed>, <DTX>, <BLER>}, {...}, ...

return
structure: for return value, see the help for FetchStruct structure arguments.

6.11.7.1.8.7 Throughput

SCPI Command :

```
FETCh:SIGNaling:MEASurement:BLER:UL:THRoughput
```

class ThroughputCls

Throughput commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability indicator'
- Cell_Name: List[str]: Name of the cell providing the measured connection
- Rel_Crc_Passed: List[float]: Throughput as percentage of scheduled throughput

- Abs_Crc_Passed: List[int]: Throughput in bit/s
- Abs_Scheduled: List[int]: Scheduled throughput in bit/s

fetch() → FetchStruct

```
# SCPI: FETCh:SIGNaling:MEASurement:BLER:UL:THRoughput
value: FetchStruct = driver.signaling.measurement.bler.uplink.throughput.fetch()
```

Returns the UL throughput results of the BLER measurement. There is one set of results {...} per cell: <Reliability>, {<CellName>, <RelCRCPassed>, <AbsCRCPassed>, <AbsScheduled>}, {...}, ...

return
structure: for return value, see the help for FetchStruct structure arguments.

6.11.7.2 CqiReporting

SCPI Command :

```
ABORt:SIGNaling:MEASurement:CQIReporting
```

class CqiReportingCls

CqiReporting commands group definition. 8 total commands, 4 Subgroups, 1 group commands

abort() → None

```
# SCPI: ABORt:SIGNaling:MEASurement:CQIReporting
driver.signaling.measurement.cqiReporting.abort()
```

Stops the measurement. The measurement enters the 'RDY' state.

abort_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: ABORt:SIGNaling:MEASurement:CQIReporting
driver.signaling.measurement.cqiReporting.abort_with_opc()
```

Stops the measurement. The measurement enters the 'RDY' state.

Same as abort, but waits for the operation to complete before continuing further. Use the RsCMX_Signaling.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.measurement.cqiReporting.clone()
```

Subgroups

6.11.7.2.1 Lte

class LteCls

Lte commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.measurement.cqiReporting.lte.clone()
```

Subgroups

6.11.7.2.1.1 Cword<Cword>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.signaling.measurement.cqiReporting.lte.cword.repcap_cword_get()
driver.signaling.measurement.cqiReporting.lte.cword.repcap_cword_set(repcap.Cword.Nr1)
```

SCPI Command :

```
FETCh:SIGNaling:MEASurement:CQIReporting:LTE:CWORd<no>
```

class CwordCls

Cword commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Cword, default value after init: Cword.Nr1

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability indicator'
- Cell_Name: List[str]: Name of the cell for which the values are reported.
- Median_Cqi: List[int]: Median of the CQI values reported by the UE.
- Range_Median_Rel: List[float]: Number of received CQI values in the range (median CQI - 1) to (median CQI + 1) , as percentage of TotalNoReports.
- Range_Median_Abs: List[int]: Number of received CQI values in the range (median CQI - 1) to (median CQI + 1) , as absolute value.
- Total_No_Reports: List[int]: Total number of received CQI values.

fetch(*cword=Cword.Default*) → FetchStruct

```
# SCPI: FETCh:SIGNaling:MEASurement:CQIReporting:LTE:CWORd<no>
value: FetchStruct = driver.signaling.measurement.cqiReporting.lte.cword.
↪fetch(cword = repcap.Cword.Default)
```

Returns the statistical evaluation of the histogram of reported CQI values. There are separate commands for LTE cells and NR cells. And there is one set of results {...} per cell: <Reliability>, {<CellName>, <MedianCqi>, <RangeMedianRel>, <RangeMedianAbs>, <TotalNoReports>}, {...}, ...

param cword
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cword')

return
structure: for return value, see the help for FetchStruct structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.measurement.cqiReporting.lte.cword.clone()
```

6.11.7.2.2 Nradio

class NradioCls

Nradio commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.measurement.cqiReporting.nradio.clone()
```

Subgroups

6.11.7.2.2.1 Cword<Cword>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.signaling.measurement.cqiReporting.nradio.cword.repcap_cword_get()
driver.signaling.measurement.cqiReporting.nradio.cword.repcap_cword_set(repcap.Cword.Nr1)
```

SCPI Command :

```
FETCh:SIGNaling:MEASurement:CQIReporting:NRADio:CWORd<no>
```

class CwordCls

Cword commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Cword, default value after init: Cword.Nr1

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability indicator'

- Cell_Name: List[str]: Name of the cell for which the values are reported.
- Median_Cqi: List[int]: Median of the CQI values reported by the UE.
- Range_Median_Rel: List[float]: Number of received CQI values in the range (median CQI - 1) to (median CQI + 1) , as percentage of TotalNoReports.
- Range_Median_Abs: List[int]: Number of received CQI values in the range (median CQI - 1) to (median CQI + 1) , as absolute value.
- Total_No_Reports: List[int]: Total number of received CQI values.

fetch(*cword=Cword.Default*) → FetchStruct

```
# SCPI: FETCh:SIGNaling:MEASurement:CQIReporting:NRADio:CWORd<no>
value: FetchStruct = driver.signaling.measurement.cqiReporting.nradio.cword.
↪fetch(cword = repcap.Cword.Default)
```

Returns the statistical evaluation of the histogram of reported CQI values. There are separate commands for LTE cells and NR cells. And there is one set of results {...} per cell: <Reliability>, {<CellName>, <MedianCqi>, <RangeMedianRel>, <RangeMedianAbs>, <TotalNoReports>}, {...}, ...

param cword
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cword')

return
structure: for return value, see the help for FetchStruct structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.measurement.cqiReporting.nradio.cword.clone()
```

6.11.7.2.3 State

SCPI Command :

```
FETCh:SIGNaling:MEASurement:CQIReporting:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → State

```
# SCPI: FETCh:SIGNaling:MEASurement:CQIReporting:STATe
value: enums.State = driver.signaling.measurement.cqiReporting.state.fetch()
```

Queries the measurement state.

return
state: OFF: Measurement off, no results. RDY: Measurement finished, valid results can be available. RUN: Measurement running.

6.11.7.2.4 Trace

class TraceCls

Trace commands group definition. 4 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.measurement.cqiReporting.trace.clone()
```

Subgroups

6.11.7.2.4.1 Lte

class LteCls

Lte commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.measurement.cqiReporting.trace.lte.clone()
```

Subgroups

6.11.7.2.4.2 Cword<Cword>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.signaling.measurement.cqiReporting.trace.lte.cword.repcap_cword_get()
driver.signaling.measurement.cqiReporting.trace.lte.cword.repcap_cword_set(repcap.Cword.
↪Nr1)
```

SCPI Command :

```
FETCh:SIGNaling:MEASurement:CQIReporting:TRACe:LTE:CWORd<no>
```

class CwordCls

Cword commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Cword, default value after init: Cword.Nr1

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability indicator'
- Cell_Name: List[str]: Name of the cell for which the values are reported.

- Value: List[int]: Comma-separated list of 16 values. They indicate how often the CQI indices 0 to 15 have been reported.

fetch(*cword=Cword.Default*) → FetchStruct

```
# SCPI: FETCh:SIGNaling:MEASurement:CQIReporting:TRACe:LTE:CWORd<no>
value: FetchStruct = driver.signaling.measurement.cqiReporting.trace.lte.cword.
↪fetch(cword = repcap.Cword.Default)
```

Returns the contents of the histogram of reported CQI values. There are separate commands for LTE cells and NR cells. And there is one set of results {...} per cell: <Reliability>, {<CellName>, <Value>CQI index 0, ..., <Value>CQI index 15}, {.. .}, ...

param cword
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cword')

return
structure: for return value, see the help for FetchStruct structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.measurement.cqiReporting.trace.lte.cword.clone()
```

6.11.7.2.4.3 Ri

SCPI Command :

```
FETCh:SIGNaling:MEASurement:CQIReporting:TRACe:LTE:RI
```

class RiCls

Ri commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability indicator'
- Cell_Name: List[str]: Name of the cell for which the values are reported.
- Count: List[int]: Number of returned values.
- Value: List[int]: Comma-separated list of Count values, indicating how often the RI values 1 to Count have been reported.

fetch() → FetchStruct

```
# SCPI: FETCh:SIGNaling:MEASurement:CQIReporting:TRACe:LTE:RI
value: FetchStruct = driver.signaling.measurement.cqiReporting.trace.lte.ri.
↪fetch()
```

Returns the contents of the histogram of reported RI values. There is one set of results {... } per LTE cell: <Reliability>, {<CellName>, <Count>, <Value>RI 1, ..., <Value>RI <Count>}, {...}, ...

return
structure: for return value, see the help for FetchStruct structure arguments.

6.11.7.2.4.4 Nradio

class NradioCls

Nradio commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.measurement.cqiReporting.trace.nradio.clone()
```

Subgroups

6.11.7.2.4.5 Cword<Cword>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.signaling.measurement.cqiReporting.trace.nradio.cword.repcap_cword_get()
driver.signaling.measurement.cqiReporting.trace.nradio.cword.repcap_cword_set(repcap.
↪Cword.Nr1)
```

SCPI Command :

```
FETCh:SIGNaling:MEASurement:CQIReporting:TRACe:NRADio:CWORd<no>
```

class CwordCls

Cword commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Cword, default value after init: Cword.Nr1

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability indicator'
- Cell_Name: List[str]: Name of the cell for which the values are reported.
- Value: List[int]: Comma-separated list of 16 values. They indicate how often the CQI indices 0 to 15 have been reported.

fetch(*cword=Cword.Default*) → FetchStruct

```
# SCPI: FETCh:SIGNaling:MEASurement:CQIReporting:TRACe:NRADio:CWORd<no>
value: FetchStruct = driver.signaling.measurement.cqiReporting.trace.nradio.
↪cword.fetch(cword = repcap.Cword.Default)
```

Returns the contents of the histogram of reported CQI values. There are separate commands for LTE cells and NR cells. And there is one set of results {...} per cell: <Reliability>, {<CellName>, <Value>CQI index 0, ..., <Value>CQI index 15}, {.. .}, ...

param cword
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cword')

return
structure: for return value, see the help for FetchStruct structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.measurement.cqiReporting.trace.nradio.cword.clone()
```

6.11.7.2.4.6 Ri

SCPI Command :

```
FETCh:SIGNaling:MEASurement:CQIReporting:TRACe:NRADio:RI
```

class RiCls

Ri commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability indicator'
- Cell_Name: List[str]: Name of the cell for which the values are reported.
- Count: List[int]: Number of returned values.
- Value: List[int]: Comma-separated list of Count values, indicating how often the RI values 0 to Count-1 have been reported.

fetch() → FetchStruct

```
# SCPI: FETCh:SIGNaling:MEASurement:CQIReporting:TRACe:NRADio:RI
value: FetchStruct = driver.signaling.measurement.cqiReporting.trace.nradio.ri.
↪fetch()
```

Returns the contents of the histogram of reported RI values. There is one set of results {...} per NR cell: <Reliability>, {<CellName>, <Count>, <Value>RI 0, ..., <Value>RI <Count>-1}, {...}, ...

return
structure: for return value, see the help for FetchStruct structure arguments.

6.11.8 Nradio

class NradioCls

Nradio commands group definition. 18 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.nradio.clone()
```

Subgroups

6.11.8.1 Cell

SCPI Command :

```
DELete:SIGNaling:NRADio:CELL
```

class CellCls

Cell commands group definition. 17 total commands, 6 Subgroups, 1 group commands

delete(*cell_name: str*) → None

```
# SCPI: DELete:SIGNaling:NRADio:CELL
driver.signaling.nradio.cell.delete(cell_name = 'abc')
```

Deletes an LTE or NR cell.

param cell_name
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.nradio.cell.clone()
```

Subgroups

6.11.8.1.1 Bwp<BwParts>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.signaling.nradio.cell.bwp.repcap_bwParts_get()
driver.signaling.nradio.cell.bwp.repcap_bwParts_set(repcap.BwParts.Nr1)
```

SCPI Command :

```
DELete:SIGNaling:NRADio:CELL:BWP
```

class BwpCls

Bwp commands group definition. 6 total commands, 4 Subgroups, 1 group commands Repeated Capability: BwParts, default value after init: BwParts.Nr1

delete(*cell_name: str*, *idn: int*) → None

```
# SCPI: DELete:SIGNaling:NRADio:CELL:BWP
driver.signaling.nradio.cell.bwp.delete(cell_name = 'abc', idn = 1)
```

Deletes the bandwidth part with the BWP <Id>. You cannot delete the initial BWP (ID 0) .

param cell_name
No help available

param idn
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.nradio.cell.bwp.clone()
```

Subgroups

6.11.8.1.1.1 Csi

class CsiCls

Csi commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.nradio.cell.bwp.csi.clone()
```

Subgroups

6.11.8.1.1.2 Trs

SCPI Command :

```
DELete:SIGNaling:NRADio:CELL:BWP<bwp_id>:CSI:TRS
```

class TrsCls

Trs commands group definition. 1 total commands, 0 Subgroups, 1 group commands

delete(*cell_name: str*, *index: int*, *bwParts=BwParts.Default*) → None

```
# SCPI: DELete:SIGNaling:NRADio:CELL:BWP<bwp_id>:CSI:TRS
driver.signaling.nradio.cell.bwp.csi.trs.delete(cell_name = 'abc', index = 1,␣
↪bwParts = repcap.BwParts.Default)
```

Deletes the TRS <Index>, for BWP <bb>.

param cell_name
No help available

param index
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.11.8.1.1.3 Harq

class HarqCls

Harq commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.nradio.cell.bwp.harq.clone()
```

Subgroups

6.11.8.1.1.4 Downlink

class DownlinkCls

Downlink commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.nradio.cell.bwp.harq.downlink.clone()
```

Subgroups

6.11.8.1.1.5 User

class UserCls

User commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.nradio.cell.bwp.harq.downlink.user.clone()
```

Subgroups

6.11.8.1.1.6 Retransm

SCPI Command :

```
DELete:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:DL:USER:RETRansm
```

class RetransmCls

Retransm commands group definition. 1 total commands, 0 Subgroups, 1 group commands

delete(*cell_name: str*, *index: int*, *count: int = None*, *bwParts=BwParts.Default*) → None

```
# SCPI: DELete:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:DL:USER:RETRansm
driver.signaling.nradio.cell.bwp.harq.downlink.user.retransm.delete(cell_name =
↪'abc', index = 1, count = 1, bwParts = repcap.BwParts.Default)
```

Removes retransmissions from the retransmission configuration for user-defined DL HARQ, for BWP <bb>.

param cell_name
No help available

param index
Index of the first retransmission to be deleted. Item 1 in the GUI corresponds to Index = 0.

param count
Number of retransmissions to be deleted. The default is 1.

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.11.8.1.1.7 Uplink

class UplinkCls

Uplink commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.nradio.cell.bwp.harq.uplink.clone()
```

Subgroups

6.11.8.1.1.8 User

class UserCls

User commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.nradio.cell.bwp.harq.uplink.user.clone()
```

Subgroups

6.11.8.1.1.9 Retransm

SCPI Command :

```
DELete:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:UL:USER:RETRansm
```

class RetransmCls

Retransm commands group definition. 1 total commands, 0 Subgroups, 1 group commands

delete(*cell_name: str*, *index: int*, *count: int = None*, *bwParts=BwParts.Default*) → None

```
# SCPI: DELete:SIGNaling:NRADio:CELL:BWP<bwp_id>:HARQ:UL:USER:RETRansm
driver.signaling.nradio.cell.bwp.harq.uplink.user.retransm.delete(cell_name =
↪'abc', index = 1, count = 1, bwParts = repcap.BwParts.Default)
```

Removes retransmissions from the retransmission configuration for user-defined UL HARQ, for the initial BWP.

param cell_name
No help available

param index
Index of the first retransmission to be deleted. Item 1 in the GUI corresponds to Index = 0.

param count
Number of retransmissions to be deleted. The default is 1.

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.11.8.1.1.10 Power

class PowerCls

Power commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.nradio.cell.bwp.power.clone()
```

Subgroups

6.11.8.1.1.11 Control

class ControlCls

Control commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.nradio.cell.bwp.power.control.clone()
```

Subgroups

6.11.8.1.1.12 State

SCPI Command :

```
FETCh:SIGNaling:NRADio:CELL:BWP<bpwid>:POWer:CONTrol:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*cell_name: str*, *bwParts=BwParts.Default*) → StatePwrControl

```
# SCPI: FETCh:SIGNaling:NRADio:CELL:BWP<bpwid>:POWer:CONTrol:STATe
value: enums.StatePwrControl = driver.signaling.nradio.cell.bwp.power.control.
↪state.fetch(cell_name = 'abc', bwParts = repcap.BwParts.Default)
```

Queries whether a TPC power control procedure is running for BWP <bb>. For example, whether commanding the UE to maximum power is still ongoing or already complete.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

return
state: Ready or running

6.11.8.1.1.13 Srs

class SrsCls

Srs commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.nradio.cell.bwp.srs.clone()
```

Subgroups

6.11.8.1.1.14 CnCodebook

class CnCodebookCls

CnCodebook commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.nradio.cell.bwp.srs.cnCodebook.clone()
```

Subgroups

6.11.8.1.1.15 Resource

SCPI Command :

```
DELete:SIGNaling:NRADio:CELL:BWP<bwp_id>:SRS:CNCodebook:RESource
```

class ResourceCls

Resource commands group definition. 1 total commands, 0 Subgroups, 1 group commands

delete(*cell_name: str*, *bwParts=BwParts.Default*) → None

```
# SCPI: DELete:SIGNaling:NRADio:CELL:BWP<bwp_id>:SRS:CNCodebook:RESource
driver.signaling.nradio.cell.bwp.srs.cnCodebook.resource.delete(cell_name = 'abc
↪', bwParts = repcap.BwParts.Default)
```

Removes an SRS resource from the resource set for periodic SRS, for BWP <bb>.

param cell_name
No help available

param bwParts
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Bwp')

6.11.8.1.2 Csi

class CsiCls

Csi commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.nradio.cell.csi.clone()
```

Subgroups

6.11.8.1.2.1 Trs

SCPI Command :

```
DELete:SIGNaling:NRADio:CELL:CSI:TRS
```

class TrsCls

Trs commands group definition. 1 total commands, 0 Subgroups, 1 group commands

delete(*cell_name: str*, *index: int*) → None

```
# SCPI: DELete:SIGNaling:NRADio:CELL:CSI:TRS
driver.signaling.nradio.cell.csi.trs.delete(cell_name = 'abc', index = 1)
```

Deletes the TRS <Index>, for the initial BWP.

param cell_name
No help available

param index
No help available

6.11.8.1.3 Harq

class HarqCls

Harq commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.nradio.cell.harq.clone()
```

Subgroups

6.11.8.1.3.1 Downlink

class DownlinkCls

Downlink commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.nradio.cell.harq.downlink.clone()
```

Subgroups

6.11.8.1.3.2 User

class UserCls

User commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.nradio.cell.harq.downlink.user.clone()
```

Subgroups

6.11.8.1.3.3 Retransm

SCPI Command :

```
DELete:SIGNaling:NRADio:CELL:HARQ:DL:USER:RETRansm
```

class RetransmCls

Retransm commands group definition. 1 total commands, 0 Subgroups, 1 group commands

delete(*cell_name: str*, *index: int*, *count: int = None*) → None

```
# SCPI: DELete:SIGNaling:NRADio:CELL:HARQ:DL:USER:RETRansm
driver.signaling.nradio.cell.harq.downlink.user.retransm.delete(cell_name = 'abc
↪', index = 1, count = 1)
```

Removes retransmissions from the retransmission configuration for user-defined DL HARQ, for the initial BWP.

param cell_name
No help available

param index
Index of the first retransmission to be deleted. Item 1 in the GUI corresponds to Index = 0.

param count
Number of retransmissions to be deleted. The default is 1.

6.11.8.1.3.4 Uplink

class UplinkCls

Uplink commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.nradio.cell.harq.uplink.clone()
```

Subgroups

6.11.8.1.3.5 User

class UserCls

User commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.nradio.cell.harq.uplink.user.clone()
```

Subgroups

6.11.8.1.3.6 Retransm

SCPI Command :

```
DELete:SIGNaling:NRADio:CELL:HARQ:UL:USER:RETRansm
```

class RetransmCls

Retransm commands group definition. 1 total commands, 0 Subgroups, 1 group commands

delete(*cell_name: str*, *index: int*, *count: int = None*) → None

```
# SCPI: DELete:SIGNaling:NRADio:CELL:HARQ:UL:USER:RETRansm
driver.signaling.nradio.cell.harq.uplink.user.retransm.delete(cell_name = 'abc',
↪ index = 1, count = 1)
```

Removes retransmissions from the retransmission configuration for user-defined UL HARQ, for the initial BWP.

param cell_name
No help available

param index
Index of the first retransmission to be deleted. Item 1 in the GUI corresponds to Index = 0.

param count
Number of retransmissions to be deleted. The default is 1.

6.11.8.1.4 Power

class PowerCls

Power commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.nradio.cell.power.clone()
```

Subgroups

6.11.8.1.4.1 Control

class ControlCls

Control commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.nradio.cell.power.control.clone()
```

Subgroups

6.11.8.1.4.2 State

SCPI Command :

```
FETCh:SIGNaling:NRADio:CELL:POWer:CONTrol:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*cell_name: str*) → StatePwrControl

```
# SCPI: FETCh:SIGNaling:NRADio:CELL:POWer:CONTrol:STATe
value: enums.StatePwrControl = driver.signaling.nradio.cell.power.control.state.
↪fetch(cell_name = 'abc')
```

Queries whether a TPC power control procedure is running for the initial BWP. For example, whether commanding the UE to maximum power is still ongoing or already complete.

param cell_name
No help available

return
state: Ready or running

6.11.8.1.5 Srs

class SrsCls

Srs commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.nradio.cell.srs.clone()
```

Subgroups

6.11.8.1.5.1 CnCodebook

class CnCodebookCls

CnCodebook commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.nradio.cell.srs.cnCodebook.clone()
```

Subgroups

6.11.8.1.5.2 Resource

SCPI Command :

```
DELete:SIGNaling:NRADio:CELL:SRS:CNCodebook:RESource
```

class ResourceCls

Resource commands group definition. 1 total commands, 0 Subgroups, 1 group commands

delete(*cell_name: str*) → None

```
# SCPI: DELete:SIGNaling:NRADio:CELL:SRS:CNCodebook:RESource
driver.signaling.nradio.cell.srs.cnCodebook.resource.delete(cell_name = 'abc')
```

Removes an SRS resource from the resource set for periodic SRS, for the initial BWP.

param cell_name
No help available

6.11.8.1.6 VcCalib

class VcCalibCls

VcCalib commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.nradio.cell.vcCalib.clone()
```

Subgroups

6.11.8.1.6.1 Branch

SCPI Command :

```
FETCh:SIGNaling:NRADio:CELL:VCCalib:BRANch
```

class BranchCls

Branch commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Branch_1: float: Power at RX1 minus power at RX0 while transmitting at TX1.
- Branch_2: float: Power at RX0 minus power at RX1 while transmitting at TX0.

fetch(*cell_name: str*) → FetchStruct

```
# SCPI: FETCh:SIGNaling:NRADio:CELL:VCCalib:BRANch
value: FetchStruct = driver.signaling.nradio.cell.vcCalib.branch.fetch(cell_
↪name = 'abc')
```

Queries the isolation per branch.

param cell_name
No help available

return
structure: for return value, see the help for FetchStruct structure arguments.

6.11.8.1.6.2 Iquality

SCPI Command :

```
FETCh:SIGNaling:NRADio:CELL:VCCalib:IQUality
```

class IqualityCls

Iquality commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*cell_name: str*) → VcCalibQuantity

```
# SCPI: FETCh:SIGNaling:NRADio:CELL:VCCalib:IQUality
value: enums.VcCalibQuantity = driver.signaling.nradio.cell.vcCalib.iquality.
↪fetch(cell_name = 'abc')
```

Queries the isolation quality.

param cell_name
No help available

return
quantity: Good, insufficient for conformance, critically low

6.11.8.1.6.3 Isolation

class IsolationCls

Isolation commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.nradio.cell.vcCalib.isolation.clone()
```

Subgroups

6.11.8.1.6.4 State

SCPI Command :

```
FETCh:SIGNaling:NRADio:CELL:VCCalib:ISOLation:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*cell_name: str*) → StatePwrControl

```
# SCPI: FETCh:SIGNaling:NRADio:CELL:VCCalib:ISOLation:STATe
value: enums.StatePwrControl = driver.signaling.nradio.cell.vcCalib.isolation.
↪state.fetch(cell_name = 'abc')
```

Queries the state of the procedure updating the isolation quality.

param cell_name
No help available

return
state: No help available

6.11.8.1.6.5 Matrix

SCPI Command :

```
FETCh:SIGNaling:NRADio:CELL:VCCalib:MATRix
```

class MatrixCls

Matrix commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- G_11_Real: float: No parameter help available
- G_11_Imaginary: float: No parameter help available
- G_12_Real: float: No parameter help available
- G_12_Imaginary: float: No parameter help available
- G_21_Real: float: No parameter help available
- G_21_Imaginary: float: No parameter help available
- G_22_Real: float: No parameter help available

- G_22_Imaginary: float: No parameter help available

fetch(*cell_name: str*) → FetchStruct

```
# SCPI: FETCh:SIGNaling:NRADio:CELL:VCCalib:MATRix
value: FetchStruct = driver.signaling.nradio.cell.vcCalib.matrix.fetch(cell_
↪name = 'abc')
```

Queries the coefficients of the calibration matrix. There are four coefficients (g11, g12, g21, g22) . Each has a real part and an imaginary part.

param cell_name
No help available

return
structure: for return value, see the help for FetchStruct structure arguments.

6.11.8.1.6.6 State

SCPI Command :

```
FETCh:SIGNaling:NRADio:CELL:VCCalib:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*cell_name: str*) → StatePwrControl

```
# SCPI: FETCh:SIGNaling:NRADio:CELL:VCCalib:STATe
value: enums.StatePwrControl = driver.signaling.nradio.cell.vcCalib.state.
↪fetch(cell_name = 'abc')
```

Queries the state of the calibration procedure.

param cell_name
No help available

return
state: No help available

6.11.8.2 Cgroup

SCPI Command :

```
DELete:SIGNaling:NRADio:CGRoup
```

class CgroupCls

Cgroup commands group definition. 1 total commands, 0 Subgroups, 1 group commands

delete(*cell_group_name: str*) → None

```
# SCPI: DELete:SIGNaling:NRADio:CGRoup
driver.signaling.nradio.cgroup.delete(cell_group_name = 'abc')
```

Deletes an LTE or NR cell group.

param cell_group_name
No help available

6.11.9 RfChannel

SCPI Commands :

```
DELete:SIGNaling:RFCHannel
RESet:SIGNaling:RFCHannel
```

class RfChannelCls

RfChannel commands group definition. 2 total commands, 0 Subgroups, 2 group commands

delete(*cell_name: str*) → None

```
# SCPI: DELete:SIGNaling:RFCHannel
driver.signaling.rfChannel.delete(cell_name = 'abc')
```

No command help available

param cell_name
No help available

reset() → None

```
# SCPI: RESet:SIGNaling:RFCHannel
driver.signaling.rfChannel.reset()
```

No command help available

reset_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: RESet:SIGNaling:RFCHannel
driver.signaling.rfChannel.reset_with_opc()
```

No command help available

Same as reset, but waits for the operation to complete before continuing further. Use the RsCMX_Signaling.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

6.11.10 Sms

SCPI Command :

```
CLEar:SIGNaling:SMS
```

class SmsCls

Sms commands group definition. 1 total commands, 0 Subgroups, 1 group commands

clear() → None

```
# SCPI: CLEar:SIGNaling:SMS
driver.signaling.sms.clear()
```

Clears information about received mobile-originated short messages.

clear_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: CLEar:SIGNaling:SMS
driver.signaling.sms.clear_with_opc()
```

Clears information about received mobile-originated short messages.

Same as clear, but waits for the operation to complete before continuing further. Use the RsCMX_Signaling.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

6.11.11 Topology

class TopologyCls

Topology commands group definition. 10 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.topology.clone()
```

Subgroups

6.11.11.1 Cnetwork

SCPI Command :

```
DELete:SIGNaling:TOPology:CNETwork
```

class CnetworkCls

Cnetwork commands group definition. 2 total commands, 1 Subgroups, 1 group commands

delete() → None

```
# SCPI: DELete:SIGNaling:TOPology:CNETwork
driver.signaling.topology.cnetwork.delete()
```

Deletes the core network.

delete_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: DELete:SIGNaling:TOPology:CNETwork
driver.signaling.topology.cnetwork.delete_with_opc()
```

Deletes the core network.

Same as delete, but waits for the operation to complete before continuing further. Use the RsCMX_Signaling.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.topology.cnetwork.clone()
```

Subgroups

6.11.11.1.1 State

SCPI Command :

```
FETCh:SIGNaling:TOPology:CNETwork:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → StateCnetwork

```
# SCPI: FETCh:SIGNaling:TOPology:CNETwork:STATe
value: enums.StateCnetwork = driver.signaling.topology.cnetwork.state.fetch()
```

Queries the state of the core network, including the states 'edit mode' and 'live mode'.

return
state: NAV: No core network available. CREating: Creating the core network. IDLE: Core network available, edit mode. TESTing: Checking whether enough resources are available. EXHausted: Not enough resources to switch to live mode. STARting: Switching from edit mode to live mode. RUNNing: Live mode. STOPping: Switching from live mode to edit mode. DELeting: Deleting the core network. ERRor: Not recoverable core network error, delete or restart the core network.

6.11.11.2 Eps

SCPI Command :

```
DELete:SIGNaling:TOPology:EPS
```

class EpsCls

Eps commands group definition. 3 total commands, 2 Subgroups, 1 group commands

delete(*name_ta_eps: str*) → None

```
# SCPI: DELete:SIGNaling:TOPology:EPS
driver.signaling.topology.eps.delete(name_ta_eps = 'abc')
```

Deletes an EPS tracking area.

param name_ta_eps
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.topology.eps.clone()
```

Subgroups

6.11.11.2.1 Bearer

SCPI Command :

```
DELete:SIGNaling:TOPology:EPS:BEARer
```

class BearerCls

Bearer commands group definition. 1 total commands, 0 Subgroups, 1 group commands

delete(*bearer_id: int*) → None

```
# SCPI: DELete:SIGNaling:TOPology:EPS:BEARer
driver.signaling.topology.eps.bearer.delete(bearer_id = 1)
```

Establishes a dedicated bearer.

param bearer_id
ID of the dedicated bearer

6.11.11.2.2 Ue

class UeCls

Ue commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.topology.eps.ue.clone()
```

Subgroups

6.11.11.2.2.1 State

SCPI Command :

```
FETCh:SIGNaling:TOPology:EPS:UE:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Emm_Reg_State_Sum: enums.RegStateB: No parameter help available
- Emm_Reg_State: enums.RegStateB: No parameter help available

fetch(*ui_id: str = None*) → FetchStruct

```
# SCPI: FETCh:SIGNaling:TOPology:EPS:UE:STATe
value: FetchStruct = driver.signaling.topology.eps.ue.state.fetch(ui_id = 'abc')
```

No command help available

param ui_id
No help available

return
structure: for return value, see the help for FetchStruct structure arguments.

6.11.11.3 Fgs

SCPI Command :

```
DELete:SIGNaling:TOPology:FGS
```

class FgsCls

Fgs commands group definition. 4 total commands, 1 Subgroups, 1 group commands

delete(*name_ta_5_g: str*) → None

```
# SCPI: DELete:SIGNaling:TOPology:FGS
driver.signaling.topology.fgs.delete(name_ta_5_g = 'abc')
```

Deletes a 5GS tracking area.

param name_ta_5_g
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.topology.fgs.clone()
```

Subgroups

6.11.11.3.1 Ue

class UeCls

Ue commands group definition. 3 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.topology.fgs.ue.clone()
```

Subgroups

6.11.11.3.1.1 Pdu

class PduCls

Pdu commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.topology.fgs.ue.pdu.clone()
```

Subgroups

6.11.11.3.1.2 QosFlow

SCPI Command :

```
DELete:SIGNaling:TOPology:FGS:UE:PDU:QOSFlow
```

class QosFlowCls

QosFlow commands group definition. 1 total commands, 0 Subgroups, 1 group commands

delete(*ue_id: str*, *pdu_session_id: int*, *qos_flow_id: int*) → None

```
# SCPI: DELete:SIGNaling:TOPology:FGS:UE:PDU:QOSFlow
driver.signaling.topology.fgs.ue.pdu.qosFlow.delete(ue_id = 'abc', pdu_session_
↪id = 1, qos_flow_id = 1)
```

Removes a QoS flow from a PDU session.

param ue_id
For future use. Enter any value.

param pdu_session_id
ID of the PDU session from which the QoS flow is removed.

param qos_flow_id
ID of the QoS flow to be removed (QFI) .

6.11.11.3.1.3 State

SCPI Command :

```
FETCh:SIGNaling:TOPology:FGS:UE:PDU:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Pdu_Session_Id_Result: List[int]: No parameter help available
- Pdu_State: List[enums.PduState]: No parameter help available

fetch(*ue_id: str = None*, *pdu_session_id: int = None*) → FetchStruct

```
# SCPI: FETCh:SIGNaling:TOPology:FGS:UE:PDU:STATe
value: FetchStruct = driver.signaling.topology.fgs.ue.pdu.state.fetch(ue_id =
↪'abc', pdu_session_id = 1)
```

No command help available

param ue_id
No help available

param pdu_session_id
No help available

return
structure: for return value, see the help for FetchStruct structure arguments.

6.11.11.3.1.4 State

SCPI Command :

```
FETCh:SIGNaling:TOPology:FGS:UE:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Mm_5_Greg_State_Sum: enums.RegState: No parameter help available
- Mm_5_Greg_State: enums.RegState: No parameter help available

fetch(*ue_id: str = None*) → FetchStruct

```
# SCPI: FETCh:SIGNaling:TOPology:FGS:UE:STATe
value: FetchStruct = driver.signaling.topology.fgs.ue.state.fetch(ue_id = 'abc')
```

No command help available

param ue_id
No help available

return
structure: for return value, see the help for FetchStruct structure arguments.

6.11.11.4 Plmn

SCPI Command :

```
DELete:SIGNaling:TOPology:PLMN
```

class PlmnCls

Plmn commands group definition. 1 total commands, 0 Subgroups, 1 group commands

delete(*name_plmn: str*) → None

```
# SCPI: DELete:SIGNaling:TOPology:PLMN
driver.signaling.topology.plmn.delete(name_plmn = 'abc')
```

Deletes a PLMN.

param name_plmn
No help available

6.11.12 Ue

class UeCls

Ue commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.signaling.ue.clone()
```

Subgroups

6.11.12.1 DcMode

SCPI Command :

```
FETCh:SIGNaling:UE:DCMode
```

class DcModeCls

DcMode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → DcMode

```
# SCPI: FETCh:SIGNaling:UE:DCMode
value: enums.DcMode = driver.signaling.ue.dcMode.fetch()
```

Queries the dual connectivity mode. Checking this mode is especially useful when setting up NSA connections, to see whether the connection setup is complete (mode EN-DC reached, connected to LTE and NR) .

return
dc_mode: No help available

6.11.12.2 Imei

SCPI Command :

```
FETCh:SIGNaling:UE:IMEI
```

class ImeiCls

Imei commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → str

```
# SCPI: FETCh:SIGNaling:UE:IMEI
value: str = driver.signaling.ue.imei.fetch()
```

No command help available

return
imei: No help available

6.11.12.3 Imsi

SCPI Command :

```
FETCh:SIGNaling:UE:IMSI
```

class ImsiCls

Imsi commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Identity_Type: enums.IdentityType: IMSI: international mobile subscriber identity NAI: network specific identifier (NSI) GCI: global cable identifier GLI: global line identifier
- Identity: str: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:SIGNaling:UE:IMSI
value: FetchStruct = driver.signaling.ue.imsi.fetch()
```

Queries the identity reported by the UE. In an EPS tracking area, the UE sends an IMSI. In a 5GS tracking area, it can also send an NSI, GCI or GLI.

return
structure: for return value, see the help for FetchStruct structure arguments.

6.11.12.4 Rcid

SCPI Command :

```
FETCh:SIGNaling:UE:RCID
```

class RcidCls

Rcid commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → RedCapId

```
# SCPI: FETCh:SIGNaling:UE:RCID
value: enums.RedCapId = driver.signaling.ue.rcid.fetch()
```

Queries the method via which the UE was identified as RedCap UE.

return
red_cap_id: - UNSPecified: UE not identified as RedCap UE - PRACh: Identification during the RACH procedure via msg1 (PRACH occasion or PRACH preamble) . - MSG3: Identification during the RACH procedure via msg3, containing the logical channel ID (LCID) . - UECap: Identification via the UE capability report.

6.11.12.5 RrcState

SCPI Command :

```
FETCh:SIGNaling:UE:RRCState
```

class RrcStateCls

RrcState commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Rrc_State: enums.RrcState: No parameter help available

- Uec_State: enums.UecState: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:SIGNaling:UE:RRCState
value: FetchStruct = driver.signaling.ue.rrcState.fetch()
```

No command help available

return
structure: for return value, see the help for FetchStruct structure arguments.

6.12 Source

class SourceCls

Source commands group definition. 3 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.clone()
```

Subgroups

6.12.1 Signaling

class SignalingCls

Signaling commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.signaling.clone()
```

Subgroups

6.12.1.1 Lte

class LteCls

Lte commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.signaling.lte.clone()
```

Subgroups

6.12.1.1.1 Cell

class CellCls

Cell commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.signaling.lte.cell.clone()
```

Subgroups

6.12.1.1.1.1 State

SCPI Command :

```
SOURce:SIGNaling:LTE:CELL:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → bool

```
# SCPI: SOURce:SIGNaling:LTE:CELL:STATe
value: bool = driver.source.signaling.lte.cell.state.get(cell_name = 'abc')
```

Turns the cell signal on or off.

param cell_name
No help available

return
state: No help available

set(*cell_name: str*, *state: bool*) → None

```
# SCPI: SOURce:SIGNaling:LTE:CELL:STATe
driver.source.signaling.lte.cell.state.set(cell_name = 'abc', state = False)
```

Turns the cell signal on or off.

param cell_name
No help available

param state
No help available

6.12.1.2 Nradio

class NradioCls

Nradio commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.signaling.nradio.clone()
```

Subgroups

6.12.1.2.1 Cell

class CellCls

Cell commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.signaling.nradio.cell.clone()
```

Subgroups

6.12.1.2.1.1 State

SCPI Command :

```
SOURce:SIGNaling:NRADio:CELL:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*cell_name: str*) → bool

```
# SCPI: SOURce:SIGNaling:NRADio:CELL:STATe
value: bool = driver.source.signaling.nradio.cell.state.get(cell_name = 'abc')
```

Turns the cell signal on or off.

param cell_name
No help available

return
state: No help available

set(*cell_name: str*, *state: bool*) → None

```
# SCPI: SOURce:SIGNaling:NRADio:CELL:STATe
driver.source.signaling.nradio.cell.state.set(cell_name = 'abc', state = False)
```

Turns the cell signal on or off.

param cell_name
No help available

param state
No help available

6.12.1.3 Topology

class TopologyCls

Topology commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.signaling.topology.clone()
```

Subgroups

6.12.1.3.1 Cnetwork

SCPI Command :

```
SOURce:SIGNaling:TOPology:CNETwork:ENABle
```

class CnetworkCls

Cnetwork commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_enable() → bool

```
# SCPI: SOURce:SIGNaling:TOPology:CNETwork:ENABle
value: bool = driver.source.signaling.topology.cnetwork.get_enable()
```

Switches between edit mode and live mode.

return
enable: ON: Switch to live mode. OFF: Switch to edit mode.

set_enable(*enable: bool*) → None

```
# SCPI: SOURce:SIGNaling:TOPology:CNETwork:ENABle
driver.source.signaling.topology.cnetwork.set_enable(enable = False)
```

Switches between edit mode and live mode.

param enable
ON: Switch to live mode. OFF: Switch to edit mode.

6.13 System

class SystemCls

System commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.clone()
```

Subgroups

6.13.1 Signaling

SCPI Command :

```
SYSTem:SIGNaling:RESet
```

class SignalingCls

Signaling commands group definition. 1 total commands, 0 Subgroups, 1 group commands

reset() → None

```
# SCPI: SYSTem:SIGNaling:RESet
driver.system.signaling.reset()
```

No command help available

reset_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: SYSTem:SIGNaling:RESet
driver.system.signaling.reset_with_opc()
```

No command help available

Same as reset, but waits for the operation to complete before continuing further. Use the RsCMX_Signaling.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

6.14 Test

class TestCls

Test commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.test.clone()
```

Subgroups

6.14.1 Signaling

class SignalingCls

Signaling commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.test.signaling.clone()
```

Subgroups

6.14.1.1 Topology

class TopologyCls

Topology commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.test.signaling.topology.clone()
```

Subgroups

6.14.1.1.1 Cnetwork

SCPI Command :

```
TEST:SIGNaling:TOPology:CNETwork
```

class CnetworkCls

Cnetwork commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: TEST:SIGNaling:TOPology:CNETwork
driver.test.signaling.topology.cnetwork.set()
```

Tests in edit mode whether switching to live mode is possible with the current configuration. To get the result of the check, use method RsCMX_Signaling.Signaling.Topology.Cnetwork.State.fetch. The check is also triggered if you try to switch to live mode via the command method RsCMX_Signaling.Source.Signaling.Topology.Cnetwork.enable.

set_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: TEST:SIGNaling:TOPology:CNETwork
driver.test.signaling.topology.cnetwork.set_with_opc()
```

Tests in edit mode whether switching to live mode is possible with the current configuration. To get the result of the check, use method RsCMX_Signaling.Signaling.Topology.Cnetwork.State.fetch. The check is also triggered if you try to switch to live mode via the command method RsCMX_Signaling.Source.Signaling.Topology.Cnetwork.enable.

Same as set, but waits for the operation to complete before continuing further. Use the RsCMX_Signaling.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

CHAPTER

SEVEN

RSCMX_SIGNALING UTILITIES

class Utilities

Common utility class. Utility functions common for all types of drivers.

Access snippet: `utils = RsCMX_Signaling.utilities`

property logger: *ScpiLogger*

Scpi Logger interface, see *here*

Access snippet: `logger = RsCMX_Signaling.utilities.logger`

property driver_version: str

Returns the instrument driver version.

property idn_string: str

Returns instrument's identification string - the response on the SCPI command *IDN?

property manufacturer: str

Returns manufacturer of the instrument.

property full_instrument_model_name: str

Returns the current instrument's full name e.g. 'FSW26'.

property instrument_model_name: str

Returns the current instrument's family name e.g. 'FSW'.

property supported_models: List[str]

Returns a list of the instrument models supported by this instrument driver.

property instrument_firmware_version: str

Returns instrument's firmware version.

property instrument_serial_number: str

Returns instrument's serial_number.

query_opc(*timeout: int = 0*) → int

SCPI command: *OPC? Queries the instrument's OPC bit and hence it waits until the instrument reports operation complete. If you define timeout > 0, the VISA timeout is set to that value just for this method call.

property instrument_status_checking: bool

Sets / returns Instrument Status Checking. When True (default is True), all the driver methods and properties are sending "SYSTem:ERRor?" at the end to immediately react on error that might have occurred. We recommend to keep the state checking ON all the time. Switch it OFF only in rare cases when you require maximum speed. The default state after initializing the session is ON.

property encoding: str

Returns string<=>bytes encoding of the session.

property opc_query_after_write: bool

Sets / returns Instrument *OPC? query sending after each command write. When True, (default is False) the driver sends *OPC? every time a write command is performed. Use this if you want to make sure your sequence is performed command-after-command.

property bin_float_numbers_format: BinFloatFormat

Sets / returns format of float numbers when transferred as binary data.

property bin_int_numbers_format: BinIntFormat

Sets / returns format of integer numbers when transferred as binary data.

clear_status() → None

Clears instrument's status system, the session's I/O buffers and the instrument's error queue.

query_all_errors() → List[str]

Queries and clears all the errors from the instrument's error queue. The method returns list of strings as error messages. If no error is detected, the return value is None. The process is: querying 'SYSTem:ERRor?' in a loop until the error queue is empty. If you want to include the error codes, call the query_all_errors_with_codes()

query_all_errors_with_codes() → List[Tuple[int, str]]

Queries and clears all the errors from the instrument's error queue. The method returns list of tuples (code: int, message: str). If no error is detected, the return value is None. The process is: querying 'SYSTem:ERRor?' in a loop until the error queue is empty.

property instrument_options: List[str]

Returns all the instrument options. The options are sorted in the ascending order starting with K-options and continuing with B-options.

reset() → None

SCPI command: *RST Sends *RST command + calls the clear_status().

default_instrument_setup() → None

Custom steps performed at the init and at the reset().

self_test(*timeout: int = None*) → Tuple[int, str]

SCPI command: *TST? Performs instrument's self-test. Returns tuple (code:int, message: str). Code 0 means the self-test passed. You can define the custom timeout in milliseconds. If you do not define it, the default selftest timeout is used (usually 60 secs).

is_connection_active() → bool

Returns true, if the VISA connection is active and the communication with the instrument still works.

reconnect(*force_close: bool = False*) → bool

If the connection is not active, the method tries to reconnect to the device If the connection is active, and force_close is False, the method does nothing. If the connection is active, and force_close is True, the method closes, and opens the session again. Returns True, if the reconnection has been performed.

property resource_name: int

Returns the resource name used in the constructor

property opc_timeout: int

Sets / returns timeout in milliseconds for all the operations that use OPC synchronization.

property visa_timeout: int

Sets / returns visa IO timeout in milliseconds.

property data_chunk_size: int

Sets / returns the maximum size of one block transferred during write/read operations

property visa_manufacturer: int

Returns the manufacturer of the current VISA session.

process_all_commands() → None

SCPI command: *WAI Stops further commands processing until all commands sent before *WAI have been executed.

write_str(*cmd: str*) → None

Writes the command to the instrument.

write(*cmd: str*) → None

This method is an alias to the write_str(). Writes the command to the instrument as string.

write_int(*cmd: str*, *param: int*) → None

Writes the command to the instrument followed by the integer parameter: e.g.: cmd = 'SELECT:INPUT' param = '2', result command = 'SELECT:INPUT 2'

write_int_with_opc(*cmd: str*, *param: int*, *timeout: int = None*) → None

Writes the command with OPC to the instrument followed by the integer parameter: e.g.: cmd = 'SELECT:INPUT' param = '2', result command = 'SELECT:INPUT 2' If you do not provide timeout, the method uses current opc_timeout.

write_float(*cmd: str*, *param: float*) → None

Writes the command to the instrument followed by the boolean parameter: e.g.: cmd = 'CENTER:FREQ' param = '10E6', result command = 'CENTER:FREQ 10E6'

write_float_with_opc(*cmd: str*, *param: float*, *timeout: int = None*) → None

Writes the command with OPC to the instrument followed by the boolean parameter: e.g.: cmd = 'CENTER:FREQ' param = '10E6', result command = 'CENTER:FREQ 10E6' If you do not provide timeout, the method uses current opc_timeout.

write_bool(*cmd: str*, *param: bool*) → None

Writes the command to the instrument followed by the boolean parameter: e.g.: cmd = 'OUTPUT' param = 'True', result command = 'OUTPUT ON'

write_bool_with_opc(*cmd: str*, *param: bool*, *timeout: int = None*) → None

Writes the command with OPC to the instrument followed by the boolean parameter: e.g.: cmd = 'OUTPUT' param = 'True', result command = 'OUTPUT ON' If you do not provide timeout, the method uses current opc_timeout.

query_str(*query: str*) → str

Sends the query to the instrument and returns the response as string. The response is trimmed of any trailing LF characters and has no length limit.

query(*query: str*) → str

This method is an alias to the query_str(). Sends the query to the instrument and returns the response as string. The response is trimmed of any trailing LF characters and has no length limit.

query_bool(*query: str*) → bool

Sends the query to the instrument and returns the response as boolean.

query_int(*query: str*) → int

Sends the query to the instrument and returns the response as integer.

query_float(*query: str*) → float

Sends the query to the instrument and returns the response as float.

write_str_with_opc(*cmd: str*, *timeout: int = None*) → None

Writes the opc-synced command to the instrument. If you do not provide timeout, the method uses current opc_timeout.

write_with_opc(*cmd: str*, *timeout: int = None*) → None

This method is an alias to the write_str_with_opc(). Writes the opc-synced command to the instrument. If you do not provide timeout, the method uses current opc_timeout.

query_str_with_opc(*query: str*, *timeout: int = None*) → str

Sends the opc-synced query to the instrument and returns the response as string. The response is trimmed of any trailing LF characters and has no length limit. If you do not provide timeout, the method uses current opc_timeout.

query_with_opc(*query: str*, *timeout: int = None*) → str

This method is an alias to the query_str_with_opc(). Sends the opc-synced query to the instrument and returns the response as string. The response is trimmed of any trailing LF characters and has no length limit. If you do not provide timeout, the method uses current opc_timeout.

query_bool_with_opc(*query: str*, *timeout: int = None*) → bool

Sends the opc-synced query to the instrument and returns the response as boolean. If you do not provide timeout, the method uses current opc_timeout.

query_int_with_opc(*query: str*, *timeout: int = None*) → int

Sends the opc-synced query to the instrument and returns the response as integer. If you do not provide timeout, the method uses current opc_timeout.

query_float_with_opc(*query: str*, *timeout: int = None*) → float

Sends the opc-synced query to the instrument and returns the response as float. If you do not provide timeout, the method uses current opc_timeout.

write_bin_block(*cmd: str*, *payload: bytes*) → None

Writes all the payload as binary data block to the instrument. The binary data header is added at the beginning of the transmission automatically, do not include it in the payload!!!

query_bin_block(*query: str*) → bytes

Queries binary data block to bytes. Throws an exception if the returned data was not a binary data. Returns data:bytes

query_bin_block_with_opc(*query: str*, *timeout: int = None*) → bytes

Sends a OPC-synced query and returns binary data block to bytes. If you do not provide timeout, the method uses current opc_timeout.

query_bin_or_ascii_float_list(*query: str*) → List[float]

Queries a list of floating-point numbers that can be returned in ASCII format or in binary format. - For ASCII format, the list numbers are decoded as comma-separated values. - For Binary Format, the numbers are decoded based on the property BinFloatFormat, usually float 32-bit (FORM REAL,32).

query_bin_or_ascii_float_list_with_opc(*query: str*, *timeout: int = None*) → List[float]

Sends a OPC-synced query and reads a list of floating-point numbers that can be returned in ASCII format or in binary format. - For ASCII format, the list numbers are decoded as comma-separated values. - For Binary Format, the numbers are decoded based on the property BinFloatFormat, usually float 32-bit (FORM REAL,32). If you do not provide timeout, the method uses current opc_timeout.

query_bin_or_ascii_int_list(*query: str*) → List[int]

Queries a list of floating-point numbers that can be returned in ASCII format or in binary format. - For ASCII format, the list numbers are decoded as comma-separated values. - For Binary Format, the numbers are decoded based on the property BinFloatFormat, usually float 32-bit (FORM REAL,32).

query_bin_or_ascii_int_list_with_opc(*query: str*, *timeout: int = None*) → List[int]

Sends a OPC-synced query and reads a list of floating-point numbers that can be returned in ASCII format or in binary format. - For ASCII format, the list numbers are decoded as comma-separated values. - For Binary Format, the numbers are decoded based on the property BinFloatFormat, usually float 32-bit (FORM REAL,32). If you do not provide timeout, the method uses current opc_timeout.

query_bin_block_to_file(*query: str*, *file_path: str*, *append: bool = False*) → None

Queries binary data block to the provided file. If append is False, any existing file content is discarded. If append is True, the new content is added to the end of the existing file, or if the file does not exit, it is created. Throws an exception if the returned data was not a binary data. Example for transferring a file from Instrument -> PC: query = f'MMEM:DATA? '{INSTR_FILE_PATH}'". Alternatively, use the dedicated methods for this purpose:

- `send_file_from_pc_to_instrument()`
- `read_file_from_instrument_to_pc()`

query_bin_block_to_file_with_opc(*query: str*, *file_path: str*, *append: bool = False*, *timeout: int = None*) → None

Sends a OPC-synced query and writes the returned data to the provided file. If append is False, any existing file content is discarded. If append is True, the new content is added to the end of the existing file, or if the file does not exit, it is created. Throws an exception if the returned data was not a binary data.

write_bin_block_from_file(*cmd: str*, *file_path: str*) → None

Writes data from the file as binary data block to the instrument using the provided command. Example for transferring a file from PC -> Instrument: cmd = f'MMEM:DATA '{INSTR_FILE_PATH}',". Alternatively, use the dedicated methods for this purpose:

- `send_file_from_pc_to_instrument()`
- `read_file_from_instrument_to_pc()`

send_file_from_pc_to_instrument(*source_pc_file: str*, *target_instr_file: str*) → None

SCPI Command: MMEM:DATA

Sends file from PC to the instrument

read_file_from_instrument_to_pc(*source_instr_file: str*, *target_pc_file: str*, *append_to_pc_file: bool = False*) → None

SCPI Command: MMEM:DATA?

Reads file from instrument to the PC.

Set the `append_to_pc_file` to True if you want to append the read content to the end of the existing PC file

get_last_sent_cmd() → str

Returns the last commands sent to the instrument. Only works in simulation mode

go_to_local() → None

Puts the instrument into local state.

go_to_remote() → None

Puts the instrument into remote state.

get_lock() → RLock

Returns the thread lock for the current session.

By default:

- If you create standard new RsCMX_Signaling instance with new VISA session, the session gets a new thread lock. You can assign it to other RsCMX_Signaling sessions in order to share one physical instrument with a multi-thread access.
- If you create new RsCMX_Signaling from an existing session, the thread lock is shared automatically making both instances multi-thread safe.

You can always assign new thread lock by calling `driver.utilities.assign_lock()`

assign_lock(*lock: RLock*) → None

Assigns the provided thread lock.

clear_lock()

Clears the existing thread lock, making the current session thread-independent from others that might share the current thread lock.

sync_from(*source: Utilities*) → None

Synchronises these Utils with the source.

CHAPTER

EIGHT

RSCMX_SIGNALING LOGGER

Check the usage in the Getting Started chapter *here*.

class ScpiLogger

Base class for SCPI logging

mode

Sets the logging ON or OFF. Additionally, you can set the logging ON only for errors. Possible values:

- LoggingMode.Off - logging is switched OFF
- LoggingMode.On - logging is switched ON
- LoggingMode.Errors - logging is switched ON, but only for error entries
- LoggingMode.Default - sets the logging to default - the value you have set with logger.default_mode

default_mode

Sets / returns the default logging mode. You can recall the default mode by calling the logger.mode = LoggingMode.Default.

Data Type
LoggingMode

device_name: str

Use this property to change the resource name in the log from the default Resource Name (e.g. TCPIP::192.168.2.101::INSTR) to another name e.g. 'MySigGen1'.

set_logging_target(*target*, *console_log: bool = None*, *udp_log: bool = None*) → None

Sets logging target - the target must implement write() and flush(). You can optionally set the console and UDP logging ON or OFF. This method switches the logging target global OFF.

get_logging_target()

Based on the global_mode, it returns the logging target: either the local or the global one.

set_logging_target_global(*console_log: bool = None*, *udp_log: bool = None*) → None

Sets logging target to global. The global target must be defined. You can optionally set the console and UDP logging ON or OFF.

log_to_console

Returns logging to console status.

log_to_udp

Returns logging to UDP status.

log_to_console_and_udp

Returns true, if both logging to UDP and console in are True.

info_raw(*log_entry: str*, *add_new_line: bool = True*) → None

Method for logging the raw string without any formatting.

info(*start_time: datetime*, *end_time: datetime*, *log_string_info: str*, *log_string: str*) → None

Method for logging one info entry. For binary log_string, use the info_bin()

error(*start_time: datetime*, *end_time: datetime*, *log_string_info: str*, *log_string: str*) → None

Method for logging one error entry.

set_relative_timestamp(*timestamp: datetime*) → None

If set, the further timestamps will be relative to the entered time.

set_relative_timestamp_now() → None

Sets the relative timestamp to the current time.

get_relative_timestamp() → datetime

Based on the global_mode, it returns the relative timestamp: either the local or the global one.

clear_relative_timestamp() → None

Clears the reference time, and the further logging continues with absolute times.

flush() → None

Flush all the entries.

log_status_check_ok

Sets / returns the current status of status checking OK. If True (default), the log contains logging of the status checking 'Status check: OK'. If False, the 'Status check: OK' is skipped - the log is more compact. Errors will still be logged.

clear_cached_entries() → None

Clears potential cached log entries. Cached log entries are generated when the Logging is ON, but no target has been defined yet.

set_format_string(*value: str*, *line_divider: str = '\n'*) → None

Sets new format string and line divider. If you just want to set the line divider, set the format string value=None The original format string is: `PAD_LEFT12(%START_TIME%) PAD_LEFT25(%DEVICE_NAME%) PAD_LEFT12(%DURATION%) %LOG_STRING_INFO%: %LOG_STRING%`

restore_format_string() → None

Restores the original format string and the line divider to LF

abbreviated_max_len_ascii: int

Defines the maximum length of one ASCII log entry. Default value is 200 characters.

abbreviated_max_len_bin: int

Defines the maximum length of one Binary log entry. Default value is 2048 bytes.

abbreviated_max_len_list: int

Defines the maximum length of one list entry. Default value is 100 elements.

bin_line_block_size: int

Defines number of bytes to display in one line. Default value is 16 bytes.

udp_port

Returns udp logging port.

target_auto_flushing

Returns status of the auto-flushing for the logging target.

CHAPTER

NINE

RSCMX_SIGNALING EVENTS

Check the usage in the Getting Started chapter *here*.

class Events

Common Events class. Event-related methods and properties. Here you can set all the event handlers.

property before_query_handler: Callable

Returns the handler of before_query events.

Returns
current `before_query_handler`

property before_write_handler: Callable

Returns the handler of before_write events.

Returns
current `before_write_handler`

property io_events_include_data: bool

Returns the current state of the io_events_include_data See the setter for more details.

property on_read_handler: Callable

Returns the handler of on_read events.

Returns
current `on_read_handler`

property on_write_handler: Callable

Returns the handler of on_write events.

Returns
current `on_write_handler`

sync_from(*source: Events*) → None

Synchronises these Events with the source.

CHAPTER

TEN

INDEX

INDEX

Symbols

```
[CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:POFFset:SSB,
[CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:PPPSsch:EPRE,
[CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:SEPRe:EPRE,
[CONFigure]:SIGNaling:NRADio:CELL:POWer:DL:TCELl:EPRE,
[CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:AUTO:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:AUTO:MINPower,
[CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:CINDex:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:IPPRecamble:RECeive,
[CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:LRSinr:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:MEEPcmax:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:MERMs:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:NPRecamble:RECeive,
[CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:PRSTep:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:PRTPower:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:RCMode:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:REDCap:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:SPRecamble:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:POWer:UL:ZCZConfig:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:PUCCh:FORMat:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:PUCCh:MODE:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:PUCCh:PAPR:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:PUCCh:SPRB:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:PUSCh:DTFS:MCSTable:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:PUSCh:DTFS:PTCRnti:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:PUSCh:TPRecoding:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:PUSCh:TSCHema:FSTMode:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:PUSCh:TSCHema:SUBMode:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:PUSCh:TSCHema:MODE,
[CONFigure]:SIGNaling:NRADio:CELL:RESelection:COMMon:QHYSt:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:RESelection:COMMon:RANGe:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:RESelection:MINLevel:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:RESelection:PRIority:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:RESelection:SEARch:INTP:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:RESelection:SEARch:INTQ:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:RESelection:SEARch:NINP:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:RESelection:SEARch:NINQ:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:RESelection:THResholds:LOW:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:RESelection:THResholds:LOW:NCR:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:RESelection:TIMer:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:APOint:ARFCn:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:APOint:FREQuency:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:APOint:LOCation:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:ASEMission:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:COMBined:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:COMBined:CFRequency:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:COMBined:CFRequency:MODE,
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:COMBined:LOCation:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:DL:APOint:ARFCn:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:DL:APOint:FREQuency:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:DL:APOint:LOCation:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:DL:BWIDth:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:DL:CFRequency:ENABle,
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:DL:CFRequency:MODE,
[CONFigure]:SIGNaling:NRADio:CELL:RFSettings:DL:IBWP:LOBW:ENABle,
```

A

B

C

D

E

F

G

I

L

M

P

R

S

T

U